Methods and Software Procedures for Processing Digital Data from Acoustic Herring Surveys

U. Buerkle

Biological Station,
St. Andrews, N.B. E0G 2X0

January 1987

Canadian Manuscript Report of Fisheries and Aquatic Sciences No. 1902

87-02637/

Canadian Manuscript Report of Fisheries and Aquatic Sciences

Manuscript reports contain scientific and technical information that contributes to existing knowledge but which deals with national or regional problems. Distribution is restricted to institutions or individuals located in particular regions of Canada. However, no restriction is placed on subject matter, and the series reflects the broad interests and policies of the Department of Fisheries and Oceans, namely, fisheries and aquatic sciences.

Manuscript reports may be cited as full publications. The correct citation appears above the abstract of each report. Each report is abstracted in *Aquatic Sciences and Fisheries Abstracts* and indexed in the Department's annual index to scientific and technical publications.

Numbers 1 900 in this series were issued as Manuscript Reports (Biological Series) of the Biological Board of Canada, and subsequent to 1937 when the name of the Board was changed by Act of Parliament, as Manuscript Reports (Biological Series) of the Fisheries Research Board of Canada. Numbers 901 1425 were issued as Manuscript Reports of the Fisheries Research Board of Canada. Numbers 1426 1550 were issued as Department of Fisheries and the Environment, Fisheries and Marine Service Manuscript Reports. The current series name was changed with report number 1551.

Manuscript reports are produced regionally but are numbered nationally. Requests for individual reports will be filled by the issuing establishment listed on the front cover and title page. Out-of-stock reports will be supplied for a fee by commercial agents.

Rapport manuscrit canadien des sciences halieutiques et aquatiques

Les rapports manuscrits contiennent des renseignements scientifiques et techniques qui constituent une contribution aux connaissances actuelles, mais qui traitent de problèmes nationaux ou régionaux. La distribution en est limitée aux organismes et aux personnes de régions particulières du Canada. Il n'y a aucune restriction quant au sujet; de fait, la série reflète la vaste gamme des intérêts et des politiques du ministère des Pêches et des Océans, c'est-à-dire les sciences halieutiques et aquatiques.

Les rapports manuscrits peuvent être cités comme des publications complètes. Le titre exact paraît au-dessus du résumé de chaque rapport. Les rapports manuscrits sont résumés dans la revue *Résumés des sciences aquatiques et halieutiques*, et ils sont classés dans l'index annuel des publications scientifiques et techniques du Ministère.

Les numéros 1 à 900 de cette série ont été publiés à titre de manuscrits (série biologique) de l'Office de biologie du Canada, et après le changement de la désignation de cet organisme par décret du Parlement, en 1937, ont été classés comme manuscrits (série biologique) de l'Office des recherches sur les pêcheries du Canada. Les numéros 901 à 1425 ont été publiés à titre de rapports manuscrits de l'Office des recherches sur les pêcheries du Canada. Les numéros 1426 à 1550 sont parus à titre de rapports manuscrits du Service des pêches et de la mer, ministère des Pêches et de l'Environnement. Le nom actuel de la série a été établi lors de la parution du numéro 1551.

Les rapports manuscrits sont produits à l'échelon régional, mais numérotés à l'échelon national. Les demandes de rapports seront satisfaites par l'établissement auteur dont le nom figure sur la couverture et la page du titre. Les rapports épuisés seront fournis contre rétribution par des agents commerciaux.

Canadian Manuscript Report of

Fisheries and Aquatic Sciences 1902

January 1987

METHODS AND SOFTWARE PROCEDURES FOR PROCESSING DIGITAL DATA FROM ACOUSTIC HERRING SURVEYS

by

U. Buerkle
Marine Fish Division
Biological Sciences Branch
Department of Fisheres and Oceans
St. Andrews, New Brunswick E0G 2X0
Canada

© Minister of Supply and Services Canada 1987

Cat. No. Fs 97-4/1902E ISSN 0706-6473

Correct citation for this publication:

Buerkle, U. 1987. Methods and software procedures for processing digital data from acoustic herring surveys. Can. MS. Rep. Fish. Aquat. Sci. 1902: iv + 63 p.

TABLE OF CONTENTS

Page

1. Introduction 1

2. Input Requirements 1
   2.1 Digital acoustic survey data 1
   2.2 Acoustic calibration data 1
   2.3 Cruise log records 1
   2.4 Echograms 1
   2.5 System operational parameters 1

3. Methods 2
   3.1 Prepare and edit log records 2
   3.2 Edit echograms 2
   3.3 Edit acoustic survey tapes 2
   3.4 Processing acoustic data tapes 3
      3.4.1 Calibration tapes 3
      3.4.2 Survey data tapes 5

4. Outputs, processing and interpretation. 6

5. Acknowledgements 8

6. References 8

Appendix A Tranducer calibration and calculation of equivalent ideal beam 9

Appendix B Calibration processing software 12

Appendix C Tapes verification software 19

Appendix D Echogram digitizing software by M. Power 23

Appendix E Survey data processing software 29

Appendix F Output processing software 50

Appendix G Echo plotting software 53

ABSTRACT

Buerkle, U. 1987. Methods and software procedures for processing digital data from acoustic herring surveys. Can. MS Rep. Fish. Aquat. Sci. 1902: iv + 63 p.

The methods and software procedures for processing digital acoustic data from Marine Fish Division herring surveys are described. The specific input data formats required are described for survey data, calibration data, survey log records, echograms and acoustic system operational parameters. The methods describe step-by-step procedures for editing and processing the input data and for producing geographic plots of fish schools by computer. Listings of all software, job streams and computer commands required are included.

RÉSUMÉ

Buerkle, U. 1987. Methods and software procedures for processing digital data from acoustic herring surveys. Can. MS Rep. Fish. Aquat. Sci. 1902: iv + 63 p.

Le présent document décrit les méthodes et les procédures de logiciel utilisées dans le traitement des données numériques tirées des recensements acousitques de hareng qui sont réalisés par la Division des poissons de mer. On y décrit l'organisation des données en entrée c'est-à-dire les résultats du dénombrement, les données d'étalonnage, les données des journaux de recensement, les échogrammes et les paramètres opérationnels du système acoustique. Dans la partie sur les méthodes, on décrit point par point les étapes à suivre pour mettre en forme et traiter les données en entrée et pour faire des plans géographiques sur ordinateur des bancs de poissons. Le document contient aussi des listes de tous les programmes, les files de travaux et les commandes informatiques nécessaires.

1. INTRODUCTION

The methods and procedures described were designed specifically for data from acoustic integration surveys in the NAFO Div. 4WX herring stock recorded with specific acoustic instrumentation. The method of surveying and the instruments have been described (Buerkle 1984; 1985a,b). Both require specialized skills, training and experience for successful operation and good results. The processing of data described here requires skill and experience in interpreting echograms and the normal computer skills of running sessions, building and manipulating files, writing job streams, and running programs. The processing is designed to be done by technical staff with minimal supervision.

Data inputs required are:

1. Acoustic survey data recorded on digital magnetic tapes in the format produced by the Intel data logging system developed at the Marine Ecology Lab. at the Bedford Institute of Oceanography.

2. Acoustic calibration data recorded on the Intel system with specific file structure as previously described (Buerkle 1984).

3. Acoustic survey log records consisting of time and geographic position of the survey vessel at regular intervals.

4. Echograms recorded by the survey sounder marked in time synchronization with the log records and a clock in the Intel system.

5. List of acoustic system operational parameters.

Equipment used in processing consists of the HP3000 system with associated 800BPI tape drives and large-scale plotter, a Tektronix 4050 system with a digitizing tablet, and an HP9820 computer with tape drive and plotter. The software used is system intrinsics, FORTRAN, TELETRAN, CEDIT, SPSS, PLOTDAT1 and the programs written specifically for this processing.

The outputs of the processing include a printout for each segment of acoustic data as identified by an editing procedure. Information included is the average acoustic area scattering coefficient of the segment and related statistics as well as the time and geographic position of the sample. The same information is stored in disc files for access for further processing. A variety of special outputs such as acoustic scattering in a specified depth layer, or scattering from each sound transmission can be produced with small changes in the programs.

The unique features of the method are that the acoustic data can be edited so that only echos from the fish selected are integrated and that outputs are directly related to geography. This greatly facilitates the estimation of acoustic abundance of the target species.

Although the methods and procedures as a whole are specific to a herring surveys program, the methods could be adapted to surveys on other species by small changes in equipment and programming.

2. INPUT REQUIREMENTS

2.1 Digital acoustic survey data

The Intel data logging system was designed to digitize demodulated analog output from echo sounders and record it in files on 9-track 800 BPI digital magnetic tape. The files consist of undefined length data buffers that are created by starting and stopping the Intel system during data recording. Each buffer contains the date and time of when it was recorded and a series of groups of 3-data words. The Intel system samples the input echo voltage at a rate of 100 micro sec. A group of 3 data words is recorded whenever the echo voltage is above a threshold voltage that is set by dial on the Intel system. The first word in each group of 3 is the time from the sounder transmit pulse to the echo and represents the depth of the echo. The second word is echo voltage level digitized by a 12 bit A/D converter. The third word is a second channel for recording digitized echo voltages as in a dual beam system and is not used in this application.

The format of the input data is described fully in Appendix E.

2.2 Acoustic calibration data

The survey data processing requires input on the acoustic system performance in terms of calibration regression parameters. These parameters a, b, c and d are determined by a method (Buerkle 1984) that involves production of calibration data tapes. The data format on the tapes is identical to that on the survey data tapes but the file structure and sequence is specific to the calibration tapes.

2.3 Cruise log records

Cruise log records are required to calculate the geographic position of acoustic data. The normal procedure, during a survey, is to record the position of the survey boat and the time at 15-min intervals throughout the survey in the cruise log book. The records are annotated with changes in acoustic system parameters such as change in depth scale, presence or absence of fish, changes in speed or course, etc.

2.4 Echograms

Echograms produced by the survey sounder are required to edit the digital acoustic data on magnetic tapes. The Intel acoustic system includes a clock which generates the time for the headings of the data buffers. The time is also displayed on a panel of the system as year, day, hour, min, sec; the day is the day of the year (i.e. Feb. 5 = day 36). The echograms produced by the sounder are synchronized with the Intel time by pushing the marker button on the recorder exactly when the sec display changes and labelling the line produced in the echogram with the appropriate time. Marks normally are made precisely on the hour and every 15 min, as well as when the system is started and stopped and at changes in course or speed. Echograms are also annotated with messages about unusual markings or conditions such as ship wakes, echos from other species, surface noise, etc.

2.5 System operational parameters

Operational parameters of the acoustic system

include all the units, settings and parameters that affect measurement results. The parameter list includes 12 items.

		(typical values)
1.	Configuration	211
2.	TVG function	20 logR-0dB
3.	Transmitter	1/1
4.	Band width	3 kHz
5.	Pulse duration	.3 ms
6.	Paper speed	75 lines/cm
7.	Depth range	0-250 m
8.	Demodulator gain	+4 dB
9.	Calibration a, b, c, d	-31.2, .975, 21.1, .1432
10.	Receive response	-81.4 dB
11.	Source level	119.1
12.	Equivalent ideal beam	.0195

The configuration (item 1) is a code indicating which system components are used and is determined as in system calibration (Buerkle 1984). Echo sounder and demodulator setting recorded are items 2-8. The calibration parameters a, b, c and d (item 9) are results of processing calibration tapes. Items 10-12 are results of transducer calibration (Appendix A).

3. METHODS

3.1 Prepare and edit log records

The time and position parameters from the cruise log book are keypunched and formatted as 10 words: day, hour, min, sec, lat. deg., lat. min, lat. sec, long. deg., long. min, long. sec: FORMAT (10I3). A listing of these data is examined to detect and correct obviously wrong entries such as time not increasing in sequence or positions miles off course. The file is kept under cruise POS.DATA where cruise is a 4-character code to denote ship and year, e.g. P085POS.DATA in the position file for the 1985 E.E. Prince survey.

3.2 Edit echograms

The aim of the edit is to produce a file of time and depth specifications to be referenced by the data processing software and to limit processing to the data in the specified time and depth windows. The procedure is to frame fish echo traces on all the echograms of the survey in sequence by moving the pointer of the Tektronix 4050 digitizer around fish traces in sequential order.

Herring traces in the echograms consist of dark areas that show transects through fish aggregations. When the traces are uniform in density and depth range along the transect, the whole transect is included as one data segment in one edit frame. When the trace is not uniform, the transect is split into shorter more homogeneous data segments by enclosing code segment in a separate edit frame.

A detailed description of the editing process follows.

1. Start up the Tektronix 4050 system as in the system user documentation.
   - use disc #6
   - use menu overlay DIGITIZE/ACOUSTIC/BOXES
   - the library is DIGITIZE
   - the program name is ACOUSTIC/BOXES
   - the user definable key functions for the program are:

     1. Show user definable key menu
     2. Initialize parameters and begin program
     3. Reset paper position
     4. Digitize boxes (continue with current data)
     5. Terminal mode sign on the HP3000
     6. Transfer data to the HP3000
     7. Read and display file status and boxes
     8. List current window boxes
     10. Digitize boxes (reinitialize array)
     20. Load program/menu to select a new program

2. Secure echogram chart in digitizer so that a time mark on the echogram on either side of the fish traces to be boxed is on the grid.

3. Press user key 2 and enter data file name e.g. P085N02
   Control passes to cursor keys
   Cursor key functions are:
   Z Digitize data
   1. Save and change paper
   2. Stop and return to menu
   3. Assign default depth
   Follow instructions displayed.
   The time and depth scales for digitizing are estabished from time marks and depth-scale marks on the echogram. The program prompts placement of the cursor and entry of the corresponding parameters.

4. To box fish data segments, place cursor in sequence at lower left, lower right, upper right and upper left corner of the proposed box. Press cursor key 2 at each position. When no signals above the fish need to be excluded from processing, the default box top (3 m) may be implemented by pressing cursor key 3 after the two bottom positions have been digitized. When an entire data segment can not be defined by a straight line across the bottom, because it is in contact with an uneven sea floor, the data is boxed in a series of narrower boxes so that the bottom of the sequence of boxes follows the sea floor contour. Overlapping the starts and ends of the boxes slightly enables the acoustic data processing software to determine that this series of boxes belong in one data segment. When all fish traces in the section of the echogram on the digitizer have been boxed, cursor key 1 saves the data. The echogram may be then repositioned to process the next section.

5. Edit data stored in the data file is transferred to the HP3000 system by UDK 5 and 6. The sequence is:

   ```
   -UDK 5
   -log on to HP3000:QEDIT/AQ
   -UDK 5
   -UDK 6 - asks for the name of file to transfer
   -cY - to get out of A mode
   -K    HP3000 file name
   - SKILL - to clear QEDITSCR
   - AQ
   - UDK 5
   - UDK 6
   - etc.
   ```

   The resulting files contain one record for each box enclosing fish signals. Each record

consists of 6 words (STARTBOX, ENDBOX, STARTBOTTOM, ENDBOTTOM, STARTTOP, ENDTOP) with FORMAT (2F12.4, 4F10.3). All records for a survey are combined in sequential order in one file, for example, P085BOX.DATA.

3.3 Edit acoustic survey tapes

Acoustic data is recorded on digital magnetic tape in files. A record of the start time and end time of files is kept in the cruise log book and is used to identify the files that contain fish data to process and those that do not. Occasionally, errors are made (or occur) so that the number and times of files on tape do not agree with the records in the log book. It is therefore necessary to identify the files on the tapes themselves. This is done by running the program ECORUNS.PROG. The program is listed in Appendix C. The file equations required are:

```
:FILE FTN15 = edit box file name, OLD

:FILE FTN10 =   tape name  ; DEV = [12]; REC =
                                        [13]
 4500,1,U, ASCII; ACC = IN
```

The program reads the time at the begining of each file. To determine the number of edit fish boxes in the last file on tape, the program requires the end time of the tape.

If the tapes of one survey are processed in reverse chronological order, the end times can be taken simply as the start time of the previously processed tape.

A sample output of the program is:

```
RUN ECORUNS.PROG


Enter tape end time (ddd,hh,mm,ss)      ?30,18,16,30

Tape  P085N05


   File  day hour min sec   fish boxes

     1 contains no data
     2   29   23  12   0      4
     3   29   23  29  50     18
     4   30    0  13   0     13
     5   30    0  49   0     14
     6   30    1  25  40     15
     7   30    2   0  20      0
     8   30    4  36   0      0
     9 contains no data
    10   30    4  58  10      0
    11   30    5  19   0      0
    12   30    5  24  50      0
    13   30    5  45  20      6
    14   30    6   8   0      5
    15   30    6  13  50      7
    16   30    6  31  20      3
Stopped processing after 20 empty files.

END OF PROGRAM
:
```

3.4 Processing acoustic data tapes

3.4.1 Calibration tapes

Calibration tapes are processed to produce the four calibration regression parameters required in survey data processing. The job stream to process a calibration tape is CALIB.JOB:

```
!JOB CALIB,USER.ACOUSTIC;TIME=?;OUTCLASS=[SP11]
                                          [LP]
!CALIB    tape name   , [12], [SP11]
                        [13]  [LP]
!EOJ
```

Where tape name is a code eg C085N01 specifying a calibration tape (C), the year (1985) and the tape number (N01).

The files required are initialized by the UDC CALIB:

```
L 21/33
  21    CALIB INPUT,DEV,OUTPUT
  22    COMMENT Sets up files to run calibration
  23    FILE FTN05 = SPSSCMDS,OLD
  24    FILE FTN06;DEV=!OUTPUT
  25    FILE FTN12 = SPSSINPT,OLD;SHR
  26    FILE FTN24 = !INPUT,OLD;DEV=!DEV;REC=4500,1,U,ASCII;ACC=IN
  27    CONTINUE
  28    RUN CALIB.PROG
  29    RESET FTN05
  30    RESET FTN06
  31    RESET FTN12
  32    RESET FTN24
  33    ###
/
```

The CALIB.PROG software (Appendix B) calls the SPSS program to calculate the calibration regression. Two files are necessary to run SPSS. The SPSSCMDS file contains the command stream for SPSS. If SPSSCMDS does not exist, it must be written in language JOB.

```
 1    RUN NAME        CALIBRATION CONFIGURATION 211
 2    VARIABLE LIST   X1 X2 X3 Y
 3    INPUT MEDIUM    DISC(SPSSINPT)
 4    INPUT FORMAT    FIXED(F8.3,F8.5,F8.3,F8.3)
 5    N OF CASES       1005
 6    REGRESSION      VARIABLES = X1 X2 X3 Y
 7                    REGRESSION = Y WITH X1 X2 X3
 8    OPTIONS         6,20
 9    READ INPUT DATA
10    FINISH
```

The SPSSINPT file receives output from CALIB.PROG and serves as input data file for SPSS. If SPSSINPT does not exist, it must be created

```
:BUILD SPSSINPT;REC=-32,8,F,ASCII;DISC=5000,32,1
```

Calibration tapes always contain 8 files, one file for each of the 8 possible equipment configurations. If only one or a few configurations were used during a survey, it is not necessary to process all 8 files. The files to be processed must be identified as an 8-digit number to replace line 1 of SPSSCMDS (in language JOB). The code number has 1's for files to process and 0's for files to skip. For example, 00100101 processes files 3, 6 and 8 only.

The tape is then mounted (on device 12 or 13) and processed by:

```
:STREAM CALIB.JOB
```

If CALIB.PROG does not exist, it can be prepared from the source by:

```
:TELETRAN CALIB.SOURCE
:PREP SOLDPASS,CALIB.PROG;CAP=PH,IA,BA;MAXDATA=32000
:SAVE CALIB.PROG
```

Output from the calibration process comes from SPSS and has the normal format for SPSS regression output as in the example (Table 1). The correspondence of regression outputs and calibration parameters is:

Regression	Calibration
Constant	a
X1	b
X2	c
X3	d

Table 1. SPSS output.

FILE NONAME (CREATION DATE = 11/ 7/85)

********** MULTIPLE REGRESSION ********** VARIABLE LIST 1
REGRESSION LIST 1

DEPENDENT VARIABLE.. Y

SUMMARY TABLE

VARIABLE	MULTIPLE R	R SQUARE	RSQ CHANGE	SIMPLE R	B	BETA
X1	.63554	.40391	.40391	.63554	.9752334	1.02634
X2	.99747	.99495	.59104	-.40115	-21.07839	-.60643
X3	.99952	.99903	.00408	-.39218	-.1431803E-01	-.26386
(CONSTANT)					-31.27504	

3.4.2 Survey data tapes

The first step in processing survey data tapes is identification of tape files that contain fish data. Processing is faster when files that contain no fish data can be skipped entirely. The files to process in each tape must be specified in the processing job stream. They are determined by comparing the time of occurrence of fish schools in the echograms with the start times of tape files on the output of the ECORUNS program.

Routine processing of survey data tape is done at night when the HP3000 system is not busy with sessions. Processing one tape may take in excess of 30000 CPU sec but normally two tapes can be processed during one night by mounting one in each of the 800BPI tape drives and streaming the second job from the first.

Files are specified by the UDC command SCHOOLS:

```
L 1/20
   1    SCHOOLS INPUT,OUTPUT1,OUTPUT2,BOXES,POSITION,DEVICE
   2    COMMENT INPUT is the name of the input tape
   3    COMMENT OUTPUT1 is the name of the output file
   4    COMMENT OUTPUT2 is the name of the output device [LP] or [SP11]
   5    COMMENT BOXES is the name of the edit box file
   6    COMMENT POSITION is the name of the positions file
   7    COMMENT DEVICE is the tape drive [12] or [13]  for input tape
   8    FILE FTN15 = !INPUT,OLD;DEV=!DEVICE;REC=4500,1,U,ASCII;ACC=IN
   9    FILE FTN19 = !OUTPUT1,OLD;REC=-128,2,F,ASCII;ACC=APPEND
  10    FILE FTN06;DEV=!OUTPUT2
  11    FILE FTN11 = !BOXES,OLD;SHR
  12    FILE FTN12 = !POSITION,OLD;SHR
  13    CONTINUE
  14    RUN SCHOOLSA.PROG
  15    RESET FTN15
  16    RESET FTN19
  17    RESET FTN11
  18    RESET FTN12
  19    RESET FTN06
  20    ///
```

The BOXES and POSITION files have been described in sections 3.3 and 3.2, respectively. The OUTPUT1 file receives one record of output for each segment of fish data processed. If it does not exist, it must be created with an appropriate name and number of records, for example:

```
:BUILD P085SUM;DISC=400;REC=-128,2,F,ASCII
```

This file is added to by each tape processed. Since tapes are not necessarily processed in chronological order, it is good practice to start with an empty output file for each night's processing.

A sample job stream for processing the first tape of a nightly set of two is:

```
1   !JOB SCHOOLS.USER.ACOUSTIC;TIME=?,OUTCLASS=LP;INPRI=7
2   !CONTINUE
3   !SCHOOLS P085N08,P085SUM,LP,P085BOX.DATA,P085POS.DATA ;2
4   31,04,20,00                 ((START PROCESSING))
5   31,04,21,41                 ((END PROCESSING))
6   P085                        ((CRUISE NUMBER))
7   3                           ((# OF FILES TO PROCESS))
8   2
9   5
10  7
11  3                           ((# OF RATE WINDOWS))
12  50,19,18,33,00,32,21,46,00  ((PULSE RATE,FROM-TO))
13  25,32,21,46,00,32,21,47,30
14  50,32,21,47,00,50,24,00,00
15  -31.75,.9734,-19.94,-.0166  ((a,b,c,d))
16  119.1                       ((SOURCE LEVEL))
17  -91.4                       ((RECEIVE RESPONSE))
18  72.8                        ((TRANSMIT RESPONSE))
19  .30                         ((PULSE LENGTH))
20  .01954                      ((EQ. IDEAL BEAM))
21  !PURGE SCHOOL01.JOB
22  !STREAM SCHOOL02.JCB
23  !EOJ
```

The parameters in the job stream are altered for different jobs most conveniently by the EDIT CHANGE command.

Line 3 calls the UDC SCHOOLS with the appropriate file names as described in the UDC listing and specifies the unit number of the tape drive used.

Lines 4 and 5 specify the day, hour, min, sec, of the start and end of the acoustic data to process.

Line 7 specifies the number of files with fish data.

Lines 8-10 specify the number of each file with fish data in consecutive lines; if more files contain fish data, lines are added as required.

Lines 11-14 specify the number of times and the times at which the pulse rate of the recording sounder was changed during recording. Pulse rate varies with the sounder depth range used, changes in depth range are recorded in the log book and in the echograms. The format of lines 12-14 is: pulse rate, start time (day, hour, min, sec) and end time (day, hour, min., sec.) on one line for each pulse rate used. All pulse rates used during the survey are normally included in every job stream to avoid having to make changes for every job.

Line 15 specifies the calibration regression parameters described in section 3.4.1.

Lines 16-20 are the system operational parameters described in section 2.5.

Lines 21-22 purge the job stream and stream the next job. These commands would be omitted in the job streams SCHOOL02.JOB that controls processing the second tape of the night.

4. OUTPUTS, PROCESSING AND INTERPRETATION

Two types of output result from the SCHOOLSA.PROG processing, 1. hardcopy printout and 2. disc file output.

An example printout is:

```
Start time                      23  16  37  25
End time                        23  16  38  33
Start Lat/Long (d.m.s.)         45 21 43        61 12 39
End Lat/Long (d.m.s.)           45 21 36        61 12 31
School length (miles)             .136
Transect course (deg.)          139.4
Number of transmissions          56
Sum of area scattering                .1017E+00
Sum of area scattering sqrd.          .2317E-03
Average area scattering               .1816E-02
Standard deviation                    .9252E-03
Coefficient of variation              .510
```

```
Standard                          Relative %
Deviations 0----+----10---+----20---+----30---+----40---+----50
     -3.5  I
     -3.0  I
     -2.5  I
     -2.0  I
     -1.5  I
     -1.0  I**************
      -.5  I****************************
       .0  I************************
       .5  I***************
      1.0  I****
      1.5  I*******
      2.0  I*****
      2.5  I****
      3.0  I**
      3.5  I
      4.0  I
End of run. EOF on input tape
Processing run   7
```

This describes one segment of acoustic data as defined by the editing process (section 3.3). Usually, each segment contains data from a single complete transect through a fish aggregation. The exception is when the editing process has split a non-homogeneous aggregation into smaller, more homogeneous segments in order to reduce variance.

A separate printout is produced for each acoustic data segment. The printouts are used only as convenient reference material and to check normality of the data. If the histogram shows a non-normal distribution of scattering levels such as a bimodal distribution, it indicates non-homogeneity in the data. The echogram should then be re-edited into smaller data segments and the data reprocessed.

Further processing consists of plotting the geographic position of acoustic data segments, calculating the areas of fish aggregations, and calculating the average area scattering coefficient and standard error for each aggregation. This processing is done using the disc file output.

The disc file output contains one record for code acoustic data segment described on the printouts. Each record contains 17 words:

	Word		Format
1.	Start time	day	I4
2.	"	hour	I4
3.	"	min	I4
4.	"	sec	I4
5.	"	ms	I4
6.	End time	day	I4
7.		hour	I4
8.		min	I4
9.		sec	I4
10.		ms	I4
11.	Start latitude		F10.5
12.	Start longitude		F10.5
13.	End Latitude		F10.5
14.	End longitude		F10.5
15.	Average area scattering coefficient		E12.6
16.	Standard deviation		E12.6
17.	Number of transmissions		I6

The records for one night's processing are automatically stored in one disc file, e.g. P085SUM. To produce one file of output in chronological order for the whole survey, each night's processing output is added to a data file, e.g. P085SUM.DATA. This file is used for plotting the geographic position of fish data segments and

calculating average area scattering coefficients for fish patches. Identification of data segments on the plots with the records in the data file is accomplished by plotting the record number with the positions. Sequential record numbers are written to bits 111 to 116 in each record by:

```
:FILE FTN12 = P085SUM.DATA, OLD
:RUN RECNUM.PROG
```

and answering program requests. A listing of RECNUM.SOURCE is in Appendix F.

The geographic position of acoustic data is plotted by the MFD program PLOTDAT1.PROG.GRNDFISH written by G. Black. Instructions for using PLOTDAT1 are documented in a manual (Black 1982) that can be referred to in detail. Plotting acoustic data however is a special case and an explanation for the calculation of the SIZE AND VIEWPORT inputs to PLOTDAT1 is necessary.

A typical job stream is:

```
 1    !JOB DICKSON.USER.ACOUSTIC;TIME=?;INPRI=2
 2    !RUN PLOTDAT1.PROG.GRNDFISH
 3    OPTION NOPRINT
 4    DEVICE 1
 5    SIZE 36.0 24.0
 6    VIEWPORT 35.5 .5 .5 23.5
 7    WINDOW 60 31 00 60 50 00 45 21 00 45 40 00
 8    AXIS .25 1 .01667 .00833
 9    NEXT .0000E+00 .0000E+00
10    TITLE 1 .5 23.5 .25 0 1 P085 DAY 4344
11    DATA 1 POS.PUB.ACOUSTIC
12    FORMAT 1 3 (T22,3I3,T13,3I3,T73,I5)
13    PLOT 1 NUMBER .0025 1 0
14    LINE 1 3 6
15    DATA 2 SUM.PUB.ACOUSTIC
16    FORMAT 2 1 (T51,F10.5,T41,F10.5,T111,I6)
17    PLOT 2 NUMBER .25 1 45
18    DATA 3 SUM.PUB.ACOUSTIC
19    FORMAT 3 1 (T51,F10.5,T41,F10.5,T111,I6)
20    PLOT 3 SYMBOL .1.1 11
21    SCALE 3 1 1 +
22    DATA 4 SUM.PUB.ACOUSTIC
23    FORMAT 4 1 (T71,F10.5,T61,F10.5,T111,I6)
24    PLOT 4 SYMBOL .1 .1 11
25    SCALE 4 1 1 +
26    XEQ
27    EXIT
28    !EOJ
/
```

Only parameters in the SIZE, VIEWPORT, WINDOW and TITLE commands need to be changed from plot to plot.

The WINDOW command (line 7) defines the geographic area to be plotted in terms of minimum and maximum longitude and latitude. The SIZE and VIEWPORT commands (lines 5 and 6) define the size of the plot. Together these commands define the scales of the X axis (longitude) and Y axis (latitude). Because the plots are used to estimate areas, it is important to have the scales in the proportion that is appropriate for the latitude of the survey. For the herring surveys at around 45°20' north, the scale ratio of latitude to longitude is 1.43.

Two examples illustrate the method of deriving the X or Y axis SIZE and VIEWPORT parameters. For a window as below, WINDOW 60 50 00 61 31 00 45 21 00 45 40 00

Axis	Maximum	Minimum	Difference
X longitude	61°31'	60°50'	41'
Y latitude	45°40'	45°21'	19'

1. To make a plot with X axis SIZE = 36.0 in and VIEWPORT .5 in smaller (=35.5 in), calculate the Y axis VIEWPORT

   35.5 x 19'/41' x 1.43 = 23.5 in

   and the Y axis SIZE is .5 in larger so that the commands are:

   ```
   SIZE 36.0   24.0
   VIEWPORT 35.5  .5  .5  23.5
   ```

2. To make a plot with Y axis SIZE = 30.0 in and VIEWPORT .5 in smaller (=29.5 in), calculate the X axis VIEWPORT

   29.5 x 41'/19' x 1/1.43 = 44.5 in

   and the X axis SIZE is .5 in larger, so the commands would be

   ```
   SIZE  45.0  30.0
   VIEWPORT  44.5  .5  .5  29.5
   ```

The other change required is in the TITLE command where the 3rd parameter (e.g. 23.5) must be the same as the Y axis VIEWPORT and the text (e.g. P085 DAY 43/44) should be appropriate.

A single plot of acoustic data normally shows the cruise track and the position of acoustic data for a small portion of the whole survey. The data to be plotted therefore are selected portions of the acoustic output data file (e.g. P085SUM.DATA) and the cruise track position file (e.g. P085POS.DATA). These data segments are stored in the files SUM and POS respectively that are accessed by PLOTDAT1. Because the P085SUM.DATA file has records of 116 bits, the SUM file must be created in language TEXT.

A sample plot is shown in Fig. 1. The dotted line is the cruise track. The crosses are the start and end of fish data segments. The start of each segment is labelled with its data record number (diagonal numbers). The perimeter of the fish patch (solid line) is drawn by eye fit to join the ends of transects through fish. The plotting is normally done on a large scale to permit separation of data segments and determination of fish boundaries. The area of the fish patch is measured in cm^2 and converted to km^2 by the scale of the plot.

The average acoustic area scattering coefficient in a fish patch is calculated from the average area scattering coefficient in each data segment in the patch weighted by the segment length. The segments to be included in the average are identified by the record numbers on the plot. The averaging is done by:

```
:FILE FTN15 = P085SUM.DATA, OLD
:RUN AVGSCATT.PROG
```

A fish patch usually includes one or more sets of sequential fish data segments, the corresponding number of record sets and their record numbers are requested by the AVGSCATT.PROG. A typical program run is:

```
:FILE FTN15 = P085SUM.DATA,OLD
:RUN AVGSCATT.PROG

 Enter number of record sets to read:   ?2

 Enter first & last record of set    1 :  ?126,157

 Enter first & last record of set    2 :  ?163,215

Avg. area scatt. coeff. and std. err. =      .6654E-03   .1808E-04
No. of schools and avg. length        =     85   .4527

END OF PROGRAM
```

Total scattering in a fish patch is the product of average area scattering coefficient and the area of the patch in m^2. Biomass is total scattering divided by the appropriate in situ target strength.

5. ACKNOWLEDGEMENTS

Development of this processing software in the early stages was greatly assisted by S. Bellis and G. Black, who developed the method to read and format Intel data. The digitizing software for the Tektronix 4050 system (Appendix D) was written by M. Power.

6. REFERENCES

Black, G. A. P. 1982. PLOTDAT 1.PROG GRNDFISH: A specialized application program for representing information geodesically. Dept. of Fisheries and Oceans, Marine Fish Division, St. Andrews, N.B.

Buerkle, U. 1984. Calibration of time-varied gain echo-sounder receivers. Can. Tech. Rep. Fish. Aquat. Sci. 1298: iii + 6 p.

Buerkle, U. 1985a. Acoustic estimation of fish abundance in a large aggregation of herring. CAFSAC Res. Doc. 85/62: 8 p.

Buerkle, U. 1985b. St. Andrews acoustic system of data collection and processing. CAFSAC Res. Doc. 85/24: 26 p.

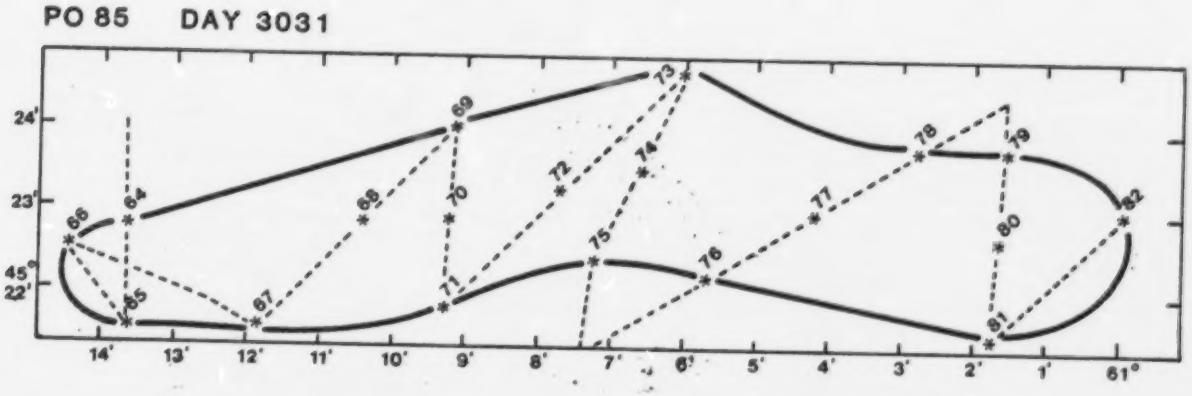

Fig. 1. Typical plot of herring patch, showing cruise track (---), start and end of fish data segments (*), the data segment record number and the fish patch boundary.

APPENDIX A

Transducer calibration and calculation of equivalent ideal beam

Transducers are calibrated by reference standard hydroplane at the Defence Research Establishment Atlantic's acoustic barge in Bedford Basin. They are calibrated in the towed body for receive response, transmit response and directivity. Directivity is shown as polar plots of intensity in decibels as a function of beam angle in degrees (Fig. 2). Source level is measured directly with the Simrad transmitters.

The equivalent ideal beam is calculated from one polar plot of one-way receive or transmit intensity or from two polar plots of receive and transmit intensity. The procedure uses an HP9820 desk-top computer disc drive and plotter.

$$\text{EQUIB} = \sum_{0}^{90} I^2(\emptyset)\, d\omega \quad \text{when}$$

$$I(\emptyset) = 10^{dB(\emptyset)/10}$$

$dB(\emptyset)$ is the intensity in decibels at half-angle $\emptyset$ from the polar plot(s).

$d\omega = 2\pi(1-\cos\emptyset)$ the increments in solid angle as $\emptyset$ increases from 0° to 90°.

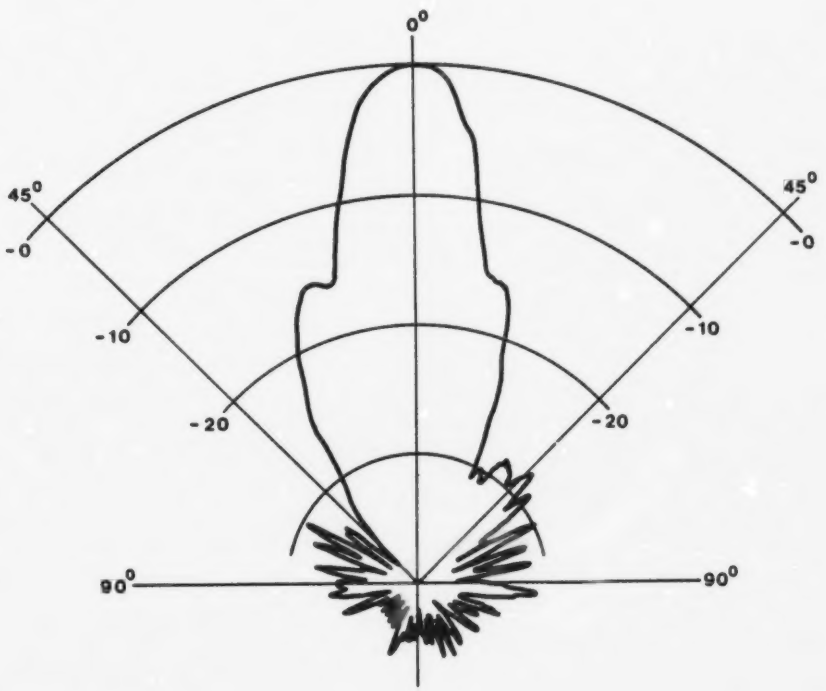

Fig. 2. Polar plot of transducer receive intensity one-way transmission.

Step by step procedure:

1. Tabulate the intensity at 1° increments of beam angles from 0° to 90° on the left and right side of one or both polar plots. The peak of the beam usually lies on the 0° axis and on a thick intensity line. This line is labelled OdB and the successive concentric thick lines are the intensities -10, -20, -30 and -40 dB. Intensities at 1° increments in beam angle can be easily read from the plot. Occasionally, when the peak does not lie on a thick intensity line, entering an OdB offset facilitates reading. The offset for each plot is the number of dB that the peak of the beam is below the nearest thick (OdB) intensity line. Offsets are entered into the HP9820 A and B parameters so that A applies to the first plot and B, if used, applies to the second plot.

2. Load cassette #8 (Beam Patterns) into the tape drive.

3. Load the program (LDF 0)

4. Run the program (RUN)

5. The program then prompts for entry of X and Y. These are the intensities from the left and right (or vice versa) side of polar plots in 1° increments from 0° to 90° off axis.

   If only one plot is used, the first pair of X and Y values are entered. In response to the next prompt for X, press RUN (this signals the program that only one plot is used) then continue entering X and Y pairs.

   If two plots are used, do not press RUN, just enter the first X and Y pair from the record plot. Then continue to enter X and Y pairs alternately from the first and second plots.

6. When 90 pairs (2 numbers) or sets of pairs (4 numbers) have been entered, the program calculates intensity for two-way transmission at 1° increments from 0° to 90°. These may be recorded in a previously marked file on the same tape

   RCF n RV, R90

   where n is the file number. This number and a file name should be recorded on the tape file list. The intensitites may also be printed by

   GO TO 24 EXECUTE RUN

7. Calculate the equivalent ideal beam (in steradians)

   GO TO 10 EXECUTE RUN

8. To plot the beam pattern function, set up the plotter and use

   ```
   SCALE    0, 90, 0, 1
   AXES     0, 0, 10, .1
   GO to 17 EXECUTE RUN
   ```

   The X axis shows the half angle Ø, the Y axis shows the proportion of total two-way transmission intensity.

APPENDIX B

Calibration processing software

```
1     $CONTROL FILE = 1-24,USLINIT,INIT.NOLIST
2           PROGRAM CALIBRATE
3     C     THIS PROGRAM IS STORED AS CALIB.SOURCE
4     C
5     C
6     C     THIS PROGRAM READS DATA FROM CALIBRATION TAPES. FINDS THE PEAK
7     C     VOLTAGE AND THE CORRESPONDING RANGE IN EACH TONE BURST OF EACH
8     C     TRANSMISSION. WRITES INPUT dBv, OUTPUT dBv AND RANGE FOR EVERY
9     C     TONE BURST IN EVERY TRANSMISSION IN EACH OF THE 14 INPUT LEVELS
10    C     IN ONE RUN TO AN OUTPUT FILE. RUNS AN SPSS REGRESSION ON THE FILE
11    C     AND GOES TO NEXT RUN FOR ALL 8 RUNS ON A TAPE
12    C
13          INTEGER # 4 CASES
14          INTEGER FNUM,CONFIGURATION(8),PIN,OUT(12),SKIP,TRANSNUM
15          INTEGER YEAR,DAY,HOUR,MIN,SEC,IN,LEN,START
16          REAL LASTPEAK,INPUT,TEMP(100,2),TIME(1497),WIDE(1497)
17          REAL NUMRUN.LIST
18          CHARACTER STRING*114
19          LOGICAL NEWRUN,SUSP,BUFFER(4500),EOF
20          CHARACTER PROG*15
21          SYSTEM INTRINSIC CREATE,ACTIVATE,QUIT,PRINTFILEINFO
22          DATA CONFIGURATION/111,112,122,121,221,222,212,211/
23    C
24    C     STEP THROUGH  RUNS TO BE PROCESSED
25    C     ACCEPT LIST OF RUNS TO BE PROCESSED IN FORM OF AN 8-DIGIT
26    C     NUMBER. 1's INDICATE TO PROCESS
27    C             0's INDICATE TO SKIP
28    C     FOR EXAMPLE 00101001 INDICATES TO PROCESS 0's 3,5 &8 ONLY
29    C
30          READ (5,8) LIST
31    C
32          NUMRUN = 10000000
33    C
34          DO I = 1,8
35    C
36             IF(LIST-NUMRUN.LT.0)
37                WRITE(6,14) CONFIGURATION(I)
38       14       FORMAT(" Bypassing cofiguration",I6)
39             IF(I.LT.8) CALL UNITCONTROL(24,3)
40                GO TO 902
41             END IF
42    C
43             LIST = LIST-NUMRUN
44             OUT12 = FNUM(12)
45             IN = FNUM(24)
46    C
47             CASES = 0
48             NEWRUN = .TRUE.
49             EOF = .FALSE.
50             SKIP = 0
51             TRANSNUM = 0
52    C
53             WRITE(6,17) CONFIGURATION(I)
```

```
      17   FORMAT(" Processing configuration",I6)
C          GET A BUFFER
C
               DO WHILE (.NOT.EOF)
                  CALL GETBUFFER(IN,BUFFER,LEN,EOF,YEAR,DAY,HOUR,MIN,SEC)
C
                  IF (.NOT. EOF)
C                    GET A TRANSMISSION
C
                     START = 4
      20             CALL GETECHO (START,LEN,BUFFER,TIME,WIDE,NUMTIMES)
                     TRANSNUM = TRANSNUM + 1
                     IF(TRANSNUM.GT.130) BREAK
C
                     IF (NEWRUN)
                        SKIP = SKIP + 1
                        IF(SKIP.LE.4) GO TO 10
                        SKIP = 0
                        TRANSNUM = TRANSNUM - 1
                        WRITE(6,16) YEAR,DAY,HOUR,MIN
      16                FORMAT(" Time of calibration",I4,I4,I3,I3)
C                       INPUT TO TEST JACK IS -20dB INTERNAL ATTENUATION
C                       IS -49 dB. TO MAKE INITIAL INPUT OF -69 dB START
C                       WITH -67 dB
                        INPUT = -67.0
                        IF(NUMTIMES.LT.5)
                           WRITE(6,18) TRANSNUM
      18                   FORMAT(" WARNING! (5 intr.  skip trnsn.",I5)
                           GO TO 10
                        END IF
                        NEWRUN = .FALSE.
                     ELSE
                        SKIP = SKIP + 1
                        IF(SKIP.GT.2.AND.SKIP.LE.10) GO TO 10
                        IF(SKIP .EQ. 11) SKIP = 1
                        IF(NUMTIMES.LT.5)
                           WRITE(6,18) TRANSNUM
                           GO TO 10
                        ELSE
                           CALL PEAK ARRAY (TIME,WIDE,TEMP,NPEAKS,LASTPEAK,
     &                                      NUMTIMES)
                           IF (SKIP .EQ. 1)
                              INPUT = INPUT-2.0
                              SAVEPEAK = LASTPEAK
                              WRITE(6,19) TRANSNUM,SAVEPEAK
      19                      FORMAT(" Transmission",I4," level",F6.1)
                              CALL WRITE FILE (TEMP,NPEAKS,INPUT,CASES)
                           ELSE
                              CALL WRITE FILE (TEMP,NPEAKS,INPUT,CASES)
                              SAVEPEAK = LASTPEAK
                           END IF
                        END IF
                     END IF
```

```
107   C
108   C
109      10          CONTINUE
110                  IF (START-LEN + 2.LE.0) GO TO 20
111               END IF
112            END DO
114   C
115            CALL UNITCONTROL(12,8)
116      905   WRITE(5@1,5000) CONFIGURATION(I)
117     5000   FORMAT('RUN NAME       CALIBRATION CONFIGURATION ',I3)
118            WRITE(5@5,5001) CASES
119     5001   FORMAT('N OF CASES     ',I5)
120            CALL UNITCONTROL(5,8)
121            PROG = 'SPSS.PUB.LIB   '
122            CALL CREATE(PROG,,PIN,,Z1L)
123            IF(.CC.) 901,900,901
124      901   CALL QUIT(99)
125      900   SUSP = Z3L
126            CALL ACTIVATE(PIN,SUSP)
127            IF(.CC.) 903,902,903
128      903   CALL QUIT(98)
129      902   CONTINUE
130   C
131            NUMRUN = NUMRUN/10
132   C
133         END DO
134   C
135      99 WRITE(6,8) ' End of tape'
136         STOP
137         END
138   C
139   C
140   C-------------------------------------------------
141         SUBROUTINE PEAK ARRAY (TIME,WIDE,TEMP,NPEAKS.LASTPEAK,NUMTIMES)
142   C-------------------------------------------------
143   C
144   C
145   C     FUNCTION: DETERMINES PEAK VOLTAGE AND CORRESPONDING RANGE
146   C     FOR EACH TONE BURST IN A TRANSMISSION. WRITES PEAK VOLTS (dBv),
147   C     AND CORRESPONDING DEPTH TO TEMP ARRAY.
148   C
149   C
150         INTEGER * 4 CASES
151         LOGICAL UP.BUFFER(4500)
152         REAL INPUT,TEMP(100,2),LASTPEAK,TIME(1497),WIDE(1497)
153   C
154   C     INITIALIZE PEAK
155   C
156         PEAK = WIDE(1)
157         UP = .FALSE.
158         NPEAKS = 0
159   C
160   C     STEP THROUGH INTERRUPTS
161   C
162         DO J = 1,NUMTIMES
163         DEPTH = TIME(J)
164   C
165   C     USE ONLY INTERRUPTS BELOW 3 METER DEPTH
166   C
167         IF (DEPTH.GT.3.0)
168   C
169            VOLTS1 = WIDE(J)
170            VOLTS2 = WIDE(J + 1)
171            IF (VOLTS2.GE.VOLTS1) UP = .TRUE.
172   C
173   C     FIND SUCCESSIVELY HIGHER PEAKS AND PUT IN TEMP ARRAY
174   C
```

```
175             IF (UP)
176                IF (VOLTS2.LT.VOLTS1)
177                   UP = .FALSE.
178                   IF (VOLTS1.GT.PEAK)
179                      NPEAKS = NPEAKS + 1
180                      PEAK = VOLTS1
181                      TEMP(NPEAKS,1) = 20.0 * ALOG10(PEAK)
182                      TEMP(NPEAKS,2) = DEPTH
183                   END IF
184                END IF
185             END IF
186             LASTPEAK = TEMP(NPEAKS,1)
187          END IF
188          END DO
189          RETURN
190          END
191    C
192    C
193    C------------------------ ------------------------------
194          SUBROUTINE WRITE FILE (TEMP,NPEAKS,INPUT,CASES)
195    C-------------------------------------------------------
196    C
197    C
198          REAL INPUT,TEMP(100,2)
199          INTEGER * 4 CASES
200    C
201          DO 40 IPEAK = 1,NPEAKS
202          PEAK = TEMP(IPEAK,1)
203          RANGE = TEMP(IPEAK,2)
204          RANGE1 = ALOG10(RANGE/3.0)
205          RANGE2 = 2.0 * (RANGE-3.0)
206          WRITE(12,24) PEAK,RANGE1,RANGE2,INPUT
207       24 FORMAT(F8.3,F8.5,F8.3,F8.3)
208          CASES = CASES +1
209       40 CONTINUE
210          RETURN
211          END
212    C
213    C-----------------------------------------------
214          SUBROUTINE GET BUFFER(IN,BUFFER,LEN,EOF,YEAR,DAY,HOUR,MIN,SEC)
215    C-------------------------------------------------------------------
216    C
217    C
218          LOGICAL BUFFER(4500),EOF
219          INTEGER IN,LEN,YEAR,DAY,HOUR,MIN,SEC
220          SYSTEM INTRINSIC FREAD,PRINTFILEINFO
221    C
222       1  LEN = FREAD(IN,BUFFER,4500)
223          IF (.CC.) 10,11,12
224       10 CALL PRINTFILEINFO(IN)
225          GO TO 1
226    C
227       12 EOF = .TRUE.
228          RETURN
229    C
230       11 CALL CHECKLEN (BUFFER,LEN)
231          CALL REVERSEBITS (BUFFER,LEN)
232          CALL GET DATE (BUFFER,YEAR,DAY,HOUR,MIN,SEC)
233    C
234          RETURN
235          END
```

```
236   C
237   C-------------------------------------------------------
238         SUBROUTINE REVERSEBITS (BUFFER,LEN)
239   C-------------------------------------------------------
240   C
241   C     THIS SUBROUTINE REVERSES THE ORDER OF ALL 8 BITS
242   C     IN EACH BYTE (2 BYTES/WORD) OF THE INPUT BUFFER.
243   C
244   C-------------------------------------------------------
245   C
246         LOGICAL TEMP,BUFFER(4500)
247   C
248         DO 20 IWORD = 1, LEN
249         TEMP(0:1)  = BUFFER(IWORD)(7:1)
250         TEMP(1:1)  = BUFFER(IWORD)(6:1)
251         TEMP(2:1)  = BUFFER(IWORD)(5:1)
252         TEMP(3:1)  = BUFFER(IWORD)(4:1)
253         TEMP(4:1)  = BUFFER(IWORD)(3:1)
254         TEMP(5:1)  = BUFFER(IWORD)(2:1)
255         TEMP(6:1)  = BUFFER(IWORD)(1:1)
256         TEMP(7:1)  = BUFFER(IWORD)(0:1)
257         TEMP(8:1)  = BUFFER(IWORD)(15:1)
258         TEMP(9:1)  = BUFFER(IWORD)(14:1)
259         TEMP(10:1) = BUFFER(IWORD)(13:1)
260         TEMP(11:1) = BUFFER(IWORD)(12:1)
261         TEMP(12:1) = BUFFER(IWORD)(11:1)
262         TEMP(13:1) = BUFFER(IWORD)(10:1)
263         TEMP(14:1) = BUFFER(IWORD)(9:1)
264         TEMP(15:1) = BUFFER(IWORD)(8:1)
265         BUFFER(IWORD) = TEMP
266       20 CONTINUE
267         RETURN
268         END
269   C-------------------------------------------------------
270         SUBROUTINE CHECKLEN (BUFFER,LEN)
271   C-------------------------------------------------------
272   C     THIS SUBROUTINE CHECKS THE LENGTH OF INPUT BUFFERS.
273   C
274   C     SUBROUTINE PARAMETERS
275         LOGICAL BUFFER(4500)
276         INTEGER LEN
277   C
278   C     LOCAL VARIABLES
279         REAL XTRANS
280   C
281   C     CHECK FOR A VALID BUFFER LENGTH
282       40 XTRANS = (LEN-3.0) / 3.0
283         IF (XTRANS.NE.AINT(XTRANS)) THEN
284           WRITE(6,42) LEN
285       42    FORMAT(' WARNING: INPUT BUFFER length of',
286      &          I5,' inappropriate.')
287         ENDIF
288         RETURN
289         END
```

```
C
C-----------------------------------------------------
      SUBROUTINE GETECHO (START,LEN,BUFFER,TIME,WIDE,NUMTIMES)
C-----------------------------------------------------
C
C     THIS SUBROUTINE CONVERTS THE BINARY CODED VALUES FOR
C     TIME AND TARGET STRENGTHS INTO INTEGER VALUES.
C
C-----------------------------------------------------
C
      INTEGER START,SKIP
      REAL TIME(1497),WIDE(1497)
      LOGICAL BUFFER(4500),TEMP1
C
      II = 0
      DO 20 J = START, LEN-2, 3
      IF (II.GE.1497) GO TO 25
      II = II + 1
      TIME(II) = INT(BUFFER(J)[4:12])
      TEMP1 = .NOT. BUFFER(J+1)
      WIDE(II)  = INT(TEMP1[4:12])
C     CONVERT A/D VALUE TO VOLTS
      WIDE(II) = WIDE(II)/409.6
C     CONVERT TIME IN 10THS OF MILLISECONDS TO DEPTH IN METERS
      TIME(II) = TIME(II) * 0.075
      IF (II-1) 10,10,30
   30 IF (TIME(II)-TIMEOLD) 40,40,10
   10 CONTINUE
      TIMEOLD = TIME(II)
   20 CONTINUE
      NUMTIMES = II
      GO TO 60
   25 WRITE(6,26)
   26 FORMAT(' WARNING: BUFFER OVERFLOW. ONLY 1497 COUNTS SAVED')
   40 NUMTIMES = II - 1
   60 START = J
      RETURN
      END
C
C-----------------------------------------------------
      SUBROUTINE GETDATE (BUFFER,YEAR,DAY,HOUR,MIN,SEC)
C-----------------------------------------------------
C
C     THIS SUBROUTINE CONVERTS THE DATE AND TIME INFORMATION
C     IN THE 1ST 6 BYTES OF A BUFFER INTO INTEGER VALUES.
C
C-----------------------------------------------------
C
      INTEGER YEAR,DAY,HOUR,MIN,SEC
      LOGICAL BUFFER(4500)
C
      YEAR = INT(BUFFER(3)[8:4])*10 + INT(BUFFER(3)[12:4])
      DAY  = INT(BUFFER(3)[4:4])*100 + INT(BUFFER(2)[8:4])*10
     &     + INT(BUFFER(2)[12:4])
      HOUR = INT(BUFFER(2)[0:4])*10 + INT(BUFFER(2)[4:4])
      MIN  = INT(BUFFER(1)[8:4])*10 + INT(BUFFER(1)[12:4])
      SEC  = INT(BUFFER(1)[0:4])*10 + INT(BUFFER(1)[4:4])
      RETURN
      END
```

APPENDIX C

Tape verification software

```
$CONTROL FILE=10-20,NOLIST
      PROGRAM RUNCOUNT
C-----------------------------------------------------------
C
C     THIS TEXT IS STORED AS ECORUMS.SOURCE.ACOUSTIC. IT IS
C     A PROGRAM TO COPY THE TIME OF THE FIRST BUFFER IN EACH
C     FILE OF INTEL TAPES AND TO DETERMINE IF THERE ARE
C     FISH IN THE FILE AS INDICATED IN THE EDIT BOXES FILE
C
C-----------------------------------------------------------
      INTEGER FNUM,FILENUM,NUMBOX,DAY,HOUR,MIN,SEC,NODATA
      INTEGER DAY1,HOUR1,MINT1,SEC1,DAY2,HOUR2,MINT2,SEC2
      LOGICAL FINISHED,BUFFER(4500),FIRST
      DOUBLE PRECISION TIME1,TIME2,TIMEEND
      SYSTEM INTRINSIC FGETINFO,FREAD,FCONTROL
      CHARACTER NAME*28,STRING
C
      DISPLAY " Enter tape end time (ddd,hh,mm,ss)     "
      ACCEPT DAY,HOUR,MIN,SEC
      CALL COMPRESSTIME (DAY,HOUR,MIN,SEC,TIMEEND)
      DISPLAY " Count fish boxes ? (Y/N)     "
      ACCEPT STRING
C
      IN=FNUM(10)
      FILENUM=1
      FIRST = .FALSE.
      NUMBOX = 0
      NODATA = 0
      CALL FGETINFO(IN,NAME)
      WRITE(6,21) NAME
   21 FORMAT(" Tape  ",S,///)
      WRITE(6,8)
    8 FORMAT("      File  day hour min sec   fish boxes"/)
   20 LEN=FREAD(IN,BUFFER,3)
      IF(.CC.) 30,40,50
   40 CONTINUE
      IF (.NOT.FIRST)
         IF(NODATA.GT.0)
            DO I = 1,NODATA,1
               WRITE(6,7) I
            END DO
            NODATA = 0
         END IF
         CALL GETTIME (BUFFER,LEN,DAY1,HOUR1,MINT1,SEC1)
         CALL COMPRESSTIME (DAY1,HOUR1,MINT1,SEC1,TIME1)
         IF(STRING.EQ."Y") CALL COUNTBOXES (TIME1,NUMBOX)
         FIRST = .TRUE.
         GO TO 100
      ELSE
         CALL GETTIME (BUFFER,LEN,DAY2,HOUR2,MINT2,SEC2)
         CALL COMPRESSTIME (DAY2,HOUR2,MINT2,SEC2,TIME2)
         IF(STRING.EQ."Y") CALL COUNTBOXES (TIME2,NUMBOX)
         CALL PRINT1 (FILENUM-NODATA,DAY1,HOUR1,MINT1,SEC1,NUMBOX)
         IF (NODATA.GT.0)
            J = FILENUM - NODATA
            DO I = J ,FILENUM-1,1
               WRITE(6,7) I
    7          FORMAT(I10," contains no data")
            END DO
            NODATA = 0
         END IF
C
         DAY1 = DAY2
         HOUR1 = HOUR2
         MINT1 = MINT2
         SEC1 = SEC2
      END IF
C
  100 CALL UNITCONTROL(10,3)
      FILENUM=FILENUM + 1
      GO TO 20
C
   50 NODATA = NODATA + 1
      IF (NODATA.EQ.20)
```

```
75              IF (FIRST)
76                 IF(STRING.EQ."Y") CALL COUNTBOXES (TIMEEND,NUMBOX)
77                 FILENUM = FILE(1) - 19
78                 CALL PRINT1 (FILENUM,DAY1,HOUR1,MINT1,SEC1,NUMBOX)
79              END IF
80              WRITE(6,8) " Stopped processing after 20 empty files."
81              STOP
82           ELSE
83              FILENUM = FILENUM + 1
84              GO TO 20
85           END IF
86     C
87     C
88        30 IF(FIRST)
89           IF(STRING.EQ."Y") CALL COUNTBOXES (TIMEEND,NUMBOX)
90           CALL PRINT1 (FILENUM-NODATA,DAY1,HOUR1,MINT1,SEC1,NUMBOX)
91           END IF
92     C
93           IF (NODATA.GT.0)
94              J = FILENUM - NODATA
95              DO I = J,FILENUM-1,1
96                 WRITE(6,7) I
97              END DO
98           END IF
99              WRITE(6,24) FILENUM
100       24    FORMAT(" Read error in file",I4,"  stop processing.")
101    C
102          STOP
103          END
105    C
106          SUBROUTINE GETTIME (BUFFER,LEN,DAY,HOUR,MIN,SEC)
107    C
108    C-------------------------------------------------------------------
109    C
110    C     THIS ROUTINE REVERSES BITS AND CALCULATES TIME FROM THE
111    C     FIRST THREE WORDS IN THE BUFFER
112    C
113    C-------------------------------------------------------------------
114    C
115          INTEGER FNUM,YEAR,DAY,HOUR,MIN,SEC,FILENUM
116          DOUBLE PRECISION TIME
117          LOGICAL BUFFER(4500),TEMP
118          DO 20 IWORD=1,LEN
119          TEMP(0:1)=BUFFER(IWORD)(7:1)
120          TEMP(1:1)=BUFFER(IWORD)(6:1)
121          TEMP(2:1)=BUFFER(IWORD)(5:1)
122          TEMP(3:1)=BUFFER(IWORD)(4:1)
123          TEMP(4:1)=BUFFER(IWORD)(3:1)
124          TEMP(5:1)=BUFFER(IWORD)(2:1)
125          TEMP(6:1)=BUFFER(IWORD)(1:1)
126          TEMP(7:1)=BUFFER(IWORD)(0:1)
127          TEMP(8:1)=BUFFER(IWORD)(15:1)
128          TEMP(9:1)=BUFFER(IWORD)(14:1)
129          TEMP(10:1)=BUFFER(IWORD)(13:1)
130          TEMP(11:1)=BUFFER(IWORD)(12:1)
131          TEMP(12:1)=BUFFER(IWORD)(11:1)
132          TEMP(13:1)=BUFFER(IWORD)(10:1)
133          TEMP(14:1)=BUFFER(IWORD)(9:1)
134          TEMP(15:1)=BUFFER(IWORD)(8:1)
135          BUFFER(IWORD)=TEMP
136       20 CONTINUE
137          YEAR=INT(BUFFER(3)(8:4))*10+INT(BUFFER(3)(12:4))
138          DAY=INT(BUFFER(3)(4:4))*100+INT(BUFFER(2)(8:4))*10
139         &   +INT(BUFFER(2)(12:4))
140          HOUR=INT(BUFFER(2)(0:4))*10+INT(BUFFER(2)(4:4))
141          MIN=INT(BUFFER(1)(8:4))*10+INT(BUFFER(1)(12:4))
142          SEC=INT(BUFFER(1)(0:4))*10+INT(BUFFER(1)(4:4))
143    C
144          RETURN
145          END
146    C
148    C
149          SUBROUTINE COMPRESSTIME (DAY,HOUR,MIN,SEC,TIME)
150    C
```

```
151   C-------------------------------------------------
152         INTEGER DAY,HOUR,MIN,SEC
153         DOUBLE PRECISION TIME
154   C
155         IF (DAY.LT.0  .OR. DAY.GT.366) GO TO 10
156         IF (HOUR.LT.0 .OR. HOUR.GT.24) GO TO 10
157         IF (MIN.LT.0  .OR. MIN.GT.60) GO TO 10
158         IF (SEC.GE.0 .AND. SEC.LE.60) GO TO 30
159   C
160      10 WRITE(6,8) '((COMPRESSTIME)) Error in time parameter'
161         WRITE(6,9) DAY,HOUR,MIN,SEC
162   C
163      30 TIME = DAY*86400.
164         TIME = TIME + HOUR*3600.
165         TIME = TIME + MIN*60.
166         TIME = TIME + SEC
167         RETURN
168         END
169   C
170   C
171   C
172   C
173   C
174   C
175   C
176   C     SUBROUTINE PRINT1 (FILENUM,DAY,HOUR,MIN,SEC,NUMBOX)
177   C
178   C-------------------------------------------------------------------
179   C
180   C     THIS ROUTINE  PRINTS THE TIME OF THE  BUFFER
181   C
182   C-------------------------------------------------------------------
183   C
184         INTEGER NUMBOX,DAY,HOUR,MIN,SEC,FILENUM,NUM
185         NUM = FILENUM-1
186         WRITE(6,21) NUM,DAY,HOUR,MIN,SEC,NUMBOX
187      21 FORMAT(I10,2I5,2I4,I7)
188         RETURN
189         END
190   C
191         SUBROUTINE COUNTBOXES (TIMESTART,NUMBOX)
192   C
193   C-----------------------------------------------
194   C
195   C     THIS SUBROUTINE COUNTS THE NUMBER OF FISH BOXES BETWEEN
196   C     START TIMES OF TAPE FILES
197   C
198   C-----------------------------------------------
199   C
200         DOUBLE PRECISION TIMESTART,STARTBOX,ENDBOX
201         INTEGER NUMBOX
202   C
203         NUMBOX = 0
204      61 READ(15,12,ERR=62,END=63) STARTBOX,ENDBOX
205      12 FORMAT(2F14.4)
206   C
207         IF (STARTBOX.LT.TIMESTART)
208            NUMBOX =    NUMBOX + 1
209            GO TO 61
210         ELSE
211            CALL UNITCONTROL(15,0)
212            GO TO 65
213         END IF
214   C
215      62 WRITE(6,8) ' READ ERROR IN BOXES FILE'
216         STOP
217      63 WRITE(6,8) 'END OF BOXES FILE'
218         STOP
219   C
220      65 RETURN
221         END
222
223
```

APPENDIX D

Echogram digitizing software

by

M. Power

```
100 REM-----------------------------------------
110 REM ssva"@PROGRAM/DIGITIZE/ACOUSTIC/BOXES";100,15000
120 REM-----------------------------------
130 D0=32
140 REM-OOK #   1    2    3    4    5    6    7    8    9   10
150 GO TO U OF 190,620,800,930,2540,3170,1860,3600,170,870
160 IF U=99 THEN 180
170 PRINT "User key ";U;" not defined.8"
180 RETURN
190 PRINT @D0:"&      PROGRAM/DIGITIZE/ACOUSTIC/BOXES
200 PRINT @D0:" A program for the digitization of acoustic sounder data.'
210 PRINT@D0:" You will be required to digitize the roll in a sequential"
220 PRINT @D0:" manner. The program is driven from the User Keys and"
230 PRINT @D0:" from the Digitizer Cursor Keys
240 PRINT @D0:"Key Function &"
250 PRINT @D0:" 1    SHOW User-Definable Key menu"
260 PRINT @D0:" 2    INITIALIZE parameters & begin program"
270 PRINT @D0:" 3    RESET paper position"
280 PRINT @D0:" 4    DIGITIZE boxes (continue with current data)"
290 PRINT @D0:" 5    TERMINAL mode to sign on the HP3000."
300 PRINT @D0:" 6    TRANSFER  data to HP3000"
310 PRINT @D0:" 7    READ and display file status & # boxes"
320 PRINT @D0:" 8    LIST current window boxes"
340 PRINT @D0:"10    DIGITIZE boxes (re-initialize array)"
350 PRINT @D0:"20    LOAD PROGRAM/MENU to select a new program."
360 PRINT @D0:"elect the key of your choice.8"
370 END
380 REM ------------------------------------------------------------------
390 REM PROBLEM 0.0 - DIGITIZE A SEQUENTIAL TIME SERIES OF BOXES OF
400 REM               WINDOWS FROM ACOUSTIC SOUNDER PAPER TRACES AND
410 REM               SAVE DATA ON HP3000 WITH ONE BOX PER RECORD.
420 REM ------------------------------------------------------------------
430 REM X(1-2)TEMPORARY VALUES FOR CALCULATION OF TIME ERRORS DIGITIZED
440 REM X(3-4),Y(3-4) DATA MIN, MAX, RANGE (VALUE), ALSO FOR WINDOW
450 REM X(5-6),Y(5-6) DATA DIGITIZER MIN. & MAX.
460 REM X(7-8),Y(7-8) PLOTTING MIN AND MAX VALUES
470 REM X(9),Y(9) DATA TRANSFORM FACTOR
480 REM X2(72,6) BOXES ARRAY WHERE MAX. OF 72 ALLOWED PER WINDOW
490 REM           X2(J,1) IS LEFT HAND TIME SLOT
500 REM           X2(J,2) IS RIGHT HAND TIME SLOT
510 REM           X2(J,3) IS LOWER LEFT (LL) DEPTH IN CM.
520 REM           Y2(J,4) IS LOWER RIGHT (LR) DEPTH IN CM.
530 REM           X2(J,5) IS UPPER RIGHT (UR) DEPTH IN CM.
540 REM           Y2(J,6) IS UPPER LEFT (UL) DEPTH IN CM.
550 REM J CURRENT ARRAY ELEMENT POINTER
560 REM T1,T2 LAST BOX START AND END TIME SLOTES
570 REM D1,D2,D3,D4 LAST BOX CORNER DEPTHS (LL,LR,UR,UL)
580 REM X3,Y3 CURRENT INPUT DIGITIZER VALUES (UNCONVERTED)
590 REM  B0 - COUNT OF CURRENT NUMBER OF BOXES ON SPECIFIED DISC FILE
600 REM  W9 - TEMP TIME VARIABLE FOR BOX TESTS
610 REM ------------------------------------------------------------------
620 INIT
630 DIM X(12),Y(12),Z1(10),B$(300),X2(72,6)
640 D0=32
650 X=0
660 Y=0
670 Z1=0
680 X2=0
690 T1=0
700 T2=0
710 D1=0
720 D2=0
730 D3=0
740 D4=0
750 X(7)=1.0E+300
760 X(8)=-1.0E+300
770 Y(7)=1.0E+300
780 Y(8)=-1.0E+300
790 GOSUB 1280
```

```
830 GOSUB 1500
840 GOSUB 1670
850 X(10)=X(3) MIN X(4)
860 Y(10)=Y(3) MIN Y(4)
870 X(11)=X(3) MAX X(4)
880 Y(11)=Y(3) MAX Y(4)
890 WINDOW X(10),X(11),Y(10),Y(11)
900 VIEWPORT 0,130,0,100
910 DELETE X2
920 DIM X2(72,6)
930 X2=0
940 J=1
950 PRINT "% BOX ARRAY INITIALIZED"
960 REM------------------BEGIN DIGITIZING-----------------
970 D0=32
972 IF X2(1,1)=0 THEN 950
974 PRINT "% BOX array now at BOX #";J
950 PRINT @D0:"%PROGRAM / DIGITIZE / ACOUSTIC / BOXES"
960 PRINT @D0:"nsure that the digitizer is in POINT mode."
970 PRINT@D0:"Program control will now be transferred to the Cursor Keys"
980 PRINT @D0:"                which will have the following functions:"
990 PRINT @D0:"                                  Z - DIGITIZE  data for box"
1000 PRINT @D0:"                                   1 - SAVE & CHANGE PAPER NOW"
1010 PRINT @D0:"                                   2 - STOP & RETURN TO MENU NOW"
1020 PRINT @D0:"                                   3 - ASSIGN default depths"
1030 PRINT @D0:"igitize now.  . . .";J
1040 REM - SET PAGING CONTROL TO AUTO
1050 PRINT @32,26;2
1060 VIEWPORT 0,130,5,50
1070 WINDOW X(5),X(6),Y(5),Y(6)
1080 REM - DIG A BOX
1090 GOSUB 2110
1100 PRINT "";
1110 REM - CHECK STATUS FLAG FOR WHAT TO DO NOW
1120 GO TO POS("8421",F$,1) OF 1170,1130,1230,1230
1130 REM - RETURN TO MENU NOW
1140 IF J=1 THEN 1160
1150 GO TO 3130
1170 REM - SAVE CURRENT WINDOW
1180 IF X2(J,1)=0 THEN 1340
1190 IF J=1 THEN 1210
1200 J=J-1
1210 GOSUB 3000
1220 GO TO POS("8421",F$,1) OF 1340,3130,1230,1230
1230 REM - CHECK BOX IN CURRENT ARRAY
1240 IF J<>1 THEN 1270
1250 J=J+1
1260 GO TO 1080
1270 GOSUB 2670
1280 REM - SAVE DATA IF ARRAY IS FULL
1290 IF J<73 THEN 1330
1300 J=J-1
1310 GOSUB 3000
1320 GO TO 860
1330 GO TO 1080
1340 REM - RESET PAPER & DATA RANGES FOR NEXT WINDOW
1350 GOSUB 1510
1360 GOSUB 1660
1370 GO TO 860
1380 REM--------SPECIFY DATA FILE-------------------------
1390 PRINT "% ENTER data file name (ie  DATA/ACOUSTIC/BOXES1)";
1400 INPUT D$
1410 D$=REP$("@",1,0)
1420 CALL "FILE",0,D$,B$
1430 IF B$<>"" THEN 1490
1440 A1=LEN(D$)
1450 A$=SEG$(D$,2,A1-1)
1460 PRINT "File ";A$;" will be created."
1470 REM - A BINARY SEQUENTIAL DISC FILE WITH 5 K BYTES WILL BE CREATED
```

```
1480 CREATE D$,"B";3889,0
1490 RETURN
1500 REM----------------SET DIGITIZING AREA---------------------
1510 PRINT "¶ Digitizing Area Specification for ACOUSTIC Data"
1520 PRI "    Set up the roll on the Graphics Tablet so that an edge"
1530 PRINT "of the paper lines up exactly with a line on the mylar "
1540 PRINT "protection sheet."
1550 PRINT "¶¶Using POINT mode, POINT to the Lower Left Corner of the"
1560 PRINT " Digitizing Area for the Data to be digitized"
1570 INPUT 98;X(5),Y(5),F$
1580 REM - CURSOR KEY NUMBER    1  -  2  -  3  -  Z
1590 GO TO POS("8421",F$,1) OF 3130,3130,3130,1600
1600 PRINT "Using POINT mode, POINT to the Upper Right Corner of the"
1610 PRINT "Digitizing Area"
1620 INPUT 98;X(6),Y(6),F$
1630 REM - CURSOR KEY NUMBER    1  -  2  -  3  -  Z
1640 GO TO POS("8421",F$,1) OF 3130,3130,3130,1650
1650 RETURN
1660 REM----------------SET DATA RANGE VALUES-----------------------
1670 PRINT "¶ Data Range Specification"
1680 PRINT"Enter the Julian day, hour, min, sec for the start position"
1690 PRINT "ie (248 18 30 00) or (248,18,30,00) (no brackets needed)¶"
1700 INPUT D,H1,M1,S1
1710 X(3)=D*86400+H1*3600+M1*60+S1
1720 PRINT "Enter the Julian day, hour, min, sec for the end position"
1730 INPUT D,H1,M1,S1
1740 X(4)=D*86400+H1*3600+M1*60+S1
1750 IF X(4)>X(3) THEN 1790
1760 PRI "ERROR. . .End time (";X(4);") is less than start (";X(3);")";
1770 PRINT "TRY AGAIN . . "
1780 GO TO 1680
1790 PRINT "Enter the Bottom & Top Values for the Depth (Metres)";
1800 INPUT Y(3),Y(4)
1810 Y(3)=Y(3)*100
1820 Y(4)=Y(4)*100
1830 X(5)=(X(4)-X(3))/(X(6)-X(5)
1840 Y(5)=(Y(4)-Y(3))/(Y(6)-Y(5)
1850 RETURN
1860 REM-------------READ A FILE FROM DISC-----------------------
1870 PRINT 932,26;0
1880 PRINT "¶ ENTER the name of the data file   ";
1890 INPUT D$
1900 D$=REP("0",1,0)
1910 CALL "FILE",0,D$,B$
1920 IF B$<>"" THEN 1960
1930 PRINT "ile ";D$;" does not exist on this disc"
1940 PRINT "lease re-enter the name of your data file.¶"
1950 GO TO 1890
1960 REM - OPEN FILE WITH READ ACCESS
1970 B0=0
1980 OPEN D$;1,"R",B$
1990 ON EOF (1) THEN 2070
2000 READ #1,J
2010 DELETE X2
2020 DIM X2(J,6)
2030 READ #1;X2
2040 B0=B0+J
2050 GOSUB 2600
2060 GO TO 2000
2070 PRINT B0;" BOXES IN FILE ";D$
2080 CLOSE
2090 END
2100 RETURN
2110 REM - DIGITIZE A BOX ------------------------------------------
2120 REM - INPUT START POSITION -lower left or cursor function -
2130 INPUT 99;X3,Y3,F$
2140 REM - CURSOR KEY NUMBER    1  -  2  -  3  -  Z
2150 GO TO POS("8421",F$,1) OF 2660,2660,2160,2160
2160 X2(J,1)=(X3-X(5))*X(9)+X(3)
```

```
2170 X2(J,3)=(Y3-Y(5))*Y(9)+Y(3)
2180 PRINT "LL= ";X2(J,1),X2(J,3)
2190 REM - MOVE X3,Y3
2200 REM -------- INPUT LOWER RIGHT POSITION
2210 INPUT @8;X3,Y3,F$
2220 X2(J,2)=(X3-X(5))*X(9)+X(3)
2230 X2(J,4)=(Y3-Y(5))*Y(9)+Y(3)
2240 IF X2(J,2)>X2(J,1) THEN 2270
2250 PRINT "ERROR. . . LR corner is NOT. Redo box.ßßß"
2260 GO TO 2110
2270 PRINT "LR= ";X2(J,2),X2(J,4)
2280 REM - DRAW X3,Y3
2290 GO TO POS("8421",F$,1) OF 2660,2660,2310,2460
2300 REM - ASSIGN DEFAULT END DEPTHS AT THE SURFACE "
2310 X2(J,5)=0
2320 X2(J,6)=0
2330 REM - RESET DEPTHS TO 0 IF NEGATIVE
2340 IF X2(J,5)>0 THEN 2370
2350 X2(J,5)=0
2360 X2(J,6)=0
2370 PRINT "UR= ";X2(J,2),X2(J,5)
2380 PRINT "UL= ";X2(J,1),X2(J,6);"%"
2390 REM - CONVERT X2 TO DIG UNITS AND DRAW BOX
2400 Y3=(X2(J,5)-Y(3))/Y(9)-Y(5)
2410 REM - DRAW X3,Y3
2420 X3=(X2(J,1)-X(3))/X(9)-X(5)
2430 Y3=(X2(J,6)-Y(3))/Y(9)-Y(5)
2440 REM - DRAW X3,Y3
2450 GO TO 2660
2460 REM -------- input upper right position
2470 INPUT @8;X3,Y3,F$
2480 GO TO POS("8421",F$,1) OF 2660,2660,2310,2490
2490 X9=(X3-X(5))*X(9)+X(3)
2500 X2(J,5)=(Y3-Y(5))*Y(9)+Y(3)
2510 IF X9>X2(J,1) AND X2(J,5)<X2(J,4) THEN 2540
2520 PRINT "ERROR. . . UR corner is NOT. Redo box.ßßß"
2530 GO TO 2110
2540 PRINT "UR= ";X2(J,2),X2(J,5)
2550 REM - DRAW X3,Y3
2560 REM -------- input upper left position
2570 INPUT @8;X3,Y3,F$
2580 GO TO POS("8421",F$,1) OF 2660,2660,2310,2590
2590 X9=(X3-X(5))*X(9)+X(3)
2600 X2(J,6)=(Y3-Y(5))*Y(9)+Y(3)
2610 IF X9<X2(J,2) AND X2(J,6)<X2(J,3) THEN 2640
2620 PRINT "ERROR. . . UL corner is NOT. Redo box.ßßß"
2630 GO TO 2110
2640 PRINT "UL= ";X2(J,1),X2(J,6);"%"
2650 REM - DRAW X3,Y3
2660 RETURN
2670 REM - TEST THIS BOX VALUES ----------------------------------------
2680 REM -TEST IF WITHIN CURRENT WINDOW AND IF SEQUENTIAL WITH LAST BOX
2690 REM -
2700 REM - CONTINUE IF SEQUENTIAL W/O OVERLAP
2710 IF X2(J,1)>=X2(J-1,2) THEN 2980
2720 IF X2(J,1)<=X2(J-1,1) THEN 2770
2730 REM - PARTIAL OVERLAP, RESET LAST END TIME
2740 X2(J-1,2)=X2(J,1)
2750 PRINT "RESET LAST BOX END TIMEßß"
2760 GO TO 2980
2770 REM - FIND POSITION IN ARRAY
2780 FOR I=1 TO J
2790 IF X2(J,1)<=X2(I,2) THEN 2810
2800 NEXT I
2810 REM - CHECK FOR OVERLAP
2820 IF X2(J,1)<=X2(I,1) THEN 2860
2830 REM - PARTIAL OVERLAP, RESET LAST BOX END TIME
2840 X2(I,2)=X2(J,1)
2850 I=I+1
```

```
2860 REM - RESET THIS BOX AT ARRAY POS I
2870 FOR K=1 TO 6
2880 X2(I,K)=X2(J,K)
2890 NEXT K
2900 REM - RESET REMAINDER OF ARRAY
2910 FOR A=I+1 TO J
2920 FOR K=1 TO 6
2930 X2(A,K)=0
2940 NEXT K
2950 NEXT A
2960 J=I
2970 PRINT "ARRAY RESET.88"
2980 J=J+1
2990 RETURN
3000 REM------------- SAVE DATA ON FILE -----------------------------
3010 CALL "FILE",0,D$,B$
3020 IF 0=POS(B$,"  0 USED",1) THEN 3050
3030 REM - ELSE DO WHAT ???
3040 REM - OPEN OLD SEQUENTIAL BINARY DISC FILE WITH UPDATE ACCESS ONLY
3050 OPEN D$;1,"U",B$
3060 REM -WRITE DATA TO DISC -
3070 DIM X2(J,6)
3080 WRITE #1;J,X2
3090 DIM X2(72,6)
3100 CLOSE
3110 PRINT "WINDOW SAVED.8"
3120 RETURN
3130 REM------------------STOP AND RETURN TO MENU-----------------
3140 U=1
3150 CLOSE
3160 GO TO 100
3170 REM----------------TRANSFER DATA ON DATA COMM. INTERFACE-------
3180 PRINT "* TRANSFER DATA TO HP3000"
3190 PRINT "It is assumed that the HP3000 is waiting to receive data"
3200 PRINT "nter the name of the file  e.g. DATA/ACOUSTIC/BOXES1"
3210 INPUT A$
3220 IF POS(A$,"@",1)=1 THEN 3240
3230 A$=REP("@",1,0)
3240 PRINT @37,0;17,255,255
3250 IMAGE 2(9D.4D,4(6D.3D))
3260 OPEN A$;1,"R",B$
3270 IF B$<>"" THEN 3300
3280 PRINT "No files found."
3290 END
3300 C$=SEG(B$,189,LEN(B$)-189)
3310 B0=0
3320 ON EOF(1) THEN 3480
3330 REM - READ IN A WINDOW
3340 READ #1;N
3350 DELETE X2
3360 DIM X2(N,6)
3370 READ #1;X2
3380 REM - WRITE OUT WINDOW TO INTERFACE
3390 PRINT @40,30;
3400 FOR I=1 TO N
3410 PRI @40; USI 3250:X2(I,1),X2(I,2),X2(I,3),X2(I,4),X2(I,5),X2(I,6)
3420 PRINT @40;
3430 PRINT @40,30;
3440 INPUT %40:D$
3450 NEXT I
3460 B0=B0+N
3470 GO TO 3330
3480 CLOSE
```

APPENDIX E

Survey data processing software

```
1     $CONTROL FILE=18-20,NOLIST,USLINIT,LOCATION,INIT,BOUNDS,NOSOURCE
2           PROGRAM ACOUST01
3     C
4     C-------------------------------------------------------------
5     C
6     C  THIS TEXT IS STORED AS SCHOOLSA.SOURCE
7     C
8     C A PROGRAM TO READ INTEL ACOUSTIC SYSTEM TAPES AND CALCULATE AVERAGE
9     C AREA SCATTERING COEFFICIENTS AND GEOGRAPHIC POSITIONS FOR INDIVIDUAL
10    C TRANSECTS THROUGH FISH SCHOOLS
11    C  FTN05 - $STDIN
12    C  FTN06 - $STDLIST
13    C  FTN15 - INPUT DATA FILE
14    C  FTN11 - TIME/DEPTH BOXES FILE
15    C  FTN12 - POSITION DATA FILE
16    C  FTN19 - OUTPUT DATA FILE
17    C
18    C FILE FTN15=(tapename);DEV=TAPE;REC=4500,1,U,ASCII;ACC=IN
19    C FILE FTN11=(boxes filename),OLD
20    C FILE FTN12=(position filename),OLD
21    C FILE FTN19 = (output file name),NEW;SAVE
22    C
23    C
24    C
25    C
26    C INPUT FORMAT:
27    C
28    C    Acoustic tapes are written by an INTEL data logger. The data is in
29    C files which are created by starting and stopping the INTEL system
30    C during data recording. The data is written in buffers of undefined
31    C length. Minimum buffer length is 3075 words. Maximum buffer length
32    C is 4500 words
33    C    Each buffer contains a time/date record ( first 3 words ) and
34    C a series of interrupt data sets.
35    C    Each interrupt data set consists of 3 words. The first word is the
36    C time after transmitt, the second word is the complement of the A/D
37    C level of the wide beam channel, the third word is the complement of
38    C the A/D level of the narrow beam channel.
39    C    The interrupt rate is 10kHz so that with continuous signal above
40    C threshhold an interrupt will be recorded every 0.1 ms.. When the time
41    C after transmitt decreases rather than increases a new transmission is
42    C indicated.
43    C    The INTEL system will store successive transmissions in a buffer
44    C until the buffer length is 3075 words. After that new transmissions go
45    C to a new buffer.
46    C    The two 8-bit bytes recorded for each word have the bits in each
47    C byte in reverse order from that on the HP3000. The order is therefore
48    C reversed by this program before other processing.
49    C
50    C    The buffer AFTER REVERSAL is organised as follows:
51    C
52    C WORD  BYTE  NIBBLE  LENGTH (BITS)    CONTENTS
53    C
```

```
C   1    1     1     4      10'ths of seconds (0-9)
C   1    1     2     4      seconds (0-9)
C   1    2     3     4      10'ths of minutes (0-5)
C   1    2     4     4      minutes (0-9)
C   2    3     5     4      10'ths of hours (0-2)
C   2    3     6     4      hours (0-9)
C   2    4     7     4      10'ths of day (0-9)
C   2    4     8     4      day (0-9)
C   3 -  5     9     4      NOT USED
C   3    5    10     4      100'ths of day (0-3)
C   3    6    11     4      10'ths of year (0-9)
C   3    6    12     4      year
C   4       bit 4    12     time of interupt 10th msec.
C   5       bit 4    12     wide beam A/D level (2.44mv.).   %
C   6       bit 4    12     narrow beam A/D level (2.44mv.). %
C   ...
C   n       bit 4    12     last time
C   n+1     bit 4    12     last wide beam   %
C   n+2     bit 4    12     last narrow beam %
C
C % Complement of value.
C
C An interupt time which is less than that of the previous
C interupt value is considered to be the start of a new transmission.
C
C
C BUFFER = THE INPUT BUFFER OF 4500 WORDS MAXIMUM
C LEN = THE LENGTH OF THE INPUT BUFFER IN WORDS
C
      INTEGER FNUM,START,CURRENTRUN,RUNS(100),CONTROL,SCHOOLTRANS
      LOGICAL BUFFER(4500),FINISHED,EOF,NEWTAPE,NOBOX,NEWRUN
      LOGICAL NEWSCHOOL,INSCHOOL
      DOUBLE PRECISION TIMESTART,TIMEEND,BUFSTART,BUFEND,TIMETRANS
      DOUBLE PRECISION SPOSTIME,EPOSTIME,STARTSCHOOL,ENDSCHOOL
      DOUBLE PRECISION RSTART(100),REND(100),STARTBOX,ENDBOX
      SYSTEM INTRINSIC FREAD
      REAL RATE(100),SALIST(4000)
C
      NEWTAPE = .TRUE.
      SPOSTIME = 0.0
C
      CALL INITIALIZE (IN,TIMESTART,TIMEEND,FINISHED,
     &          NUMIN,RUNS,CURRENTRUN,NUMRATES,RATE,
     &          RSTART,REND,IRATE,A,B,C,D,RR,TR,PULSE,BW)

C     FOR EACH OF THE RUNS TO BE PROCESSED.
      DO J = 1, NUMIN
         EOF = .FALSE.
         NEWRUN = .TRUE.
         IF (CURRENTRUN .GT. RUNS(J)) THEN
C           THE NEXT RUN CANNOT BE BEFORE THE CURRENT RUN
            WRITE(6,*) "INPUT RUN",RUNS(J),
     &              "CAN'T BE PROCESED WHILE AT RUN",
```

```
     &                CURRENTRUN," IN INPUT TAPE."
          STOP
        ENDIF

        IF (CURRENTRUN .LT. RUNS(J)) THEN
C         JUMP PAST 'EOF' MARKS IF REQUIRED
          DO K = CURRENTRUN, RUNS(J)-1
            WRITE(6,14) K
   14       FORMAT(" Bypassing Run ",I3)
            CALL UNITCONTROL (15,3)
          END DO
          CURRENTRUN = RUNS(J)
        ENDIF

        WRITE(6,17) CURRENTRUN
   17   FORMAT(" Processing run ",I3)

C
        IF (NEWTAPE)
        CALL TRACK BOXES (TIMESTART,NOBOX,STARTBOX,ENDBOX,
     &                STARTBOTTOM,ENDBOTTOM,STARTTOP,ENDTOP)
C
        CALL GET POSITION (STARTBOX,SPOSLAT,SPOSLONG,EPOSLAT,
     &                EPOSLONG,SPOSTIME,EPOSTIME)
        NEWTAPE = .FALSE.
C
        SCHOOLSIGMA = 0.0
        SCHOOLSIGMASQR = 0.0
        SCHOOLTRANS = 0
C
        END IF
C
C
        DO WHILE (.NOT.FINISHED .AND. .NOT.EOF)

          CALL GETBUFFER (BUFFER,LEN,EOF,RATE,NUMTRANS1,TIMESTART,
     &                NUMRATES,RSTART,REND,IRATE,STARTBOX,
     &                TRANSRATE,IN,BUFSTART,BUFEND)
          IF (EOF) THEN
C
          IF (INSCHOOL)
          CALL SCHOOL END(TIMETRANS,NOBOX,NEWSCHOOL,INSCHOOL,
     &          STARTBOX,ENDBOX,STARTBOTTOM,ENDBOTTOM,STARTTOP,ENDTOP,
     &          SCHOOLTRANS,SCHOOLSIGMA,SCHOOLSIGMASQR,SALIST,
     &          SPOSTIME,EPOSTIME,SPOSLAT,EPOSLAT,SPOSLONG,EPOSLONG,
     &          STARTSCHOOL,ENDSCHOOL,SSLAT,SSLONG,ESLAT,ESLONG,
     &          TIMESTART,TIMEEND,FINISHED)
          END IF
            WRITE(6,60)
   60       FORMAT(" End of run. EOF on input tape")
          ELSE
            CALL IN BUFFER (LEN,BUFFER,TIMESTART,TIMEEND,BUFSTART,
     &          BUFEND,NUMRATES,RATE,RSTART,REND,IRATE,TRANSRATE,
```

```
161         &          STARTBOX,ENDBOX,STARTBOTTOM,ENDBOTTOM,STARTTOP,ENDTOP,
162         &          SCHOOLTRANS,FINISHED,NEWRUN,
163         &          SPOSTIME,EPOSTIME,SPOSLAT,SPOSLONG,EPOSLAT,EPOSLONG,
164         &          STARTSCHOOL,ENDSCHOOL,SSLAT,SSLONG,ESLAT,ESLONG,
165         &          NOBOX,NEWSCHOOL,INSCHOOL,SALIST,
166         &          A,B,C,D,SL,RR,TR,PULSE,BW,SCHOOLSIGMA,SCHOOLSIGMASQR)
167     C
168                  IF (FINISHED) THEN
169     C
170                  IF (INSCHOOL)
171                CALL SCHOOL END(TIMETRANS,NOBOX,NEWSCHOOL,INSCHOOL,
172         &          STARTBOX,ENDBOX,STARTBOTTOM,ENDBOTTOM,STARTTOP,ENDTOP,
173         &         SCHOOLTRANS,SCHOOLSIGMA,SCHOOLSIGMASQR,SALIST,
174         &         SPOSTIME,EPOSTIME,SPOSLAT,EPOSLAT,SPOSLONG,EPOSLONG,
175         &         STARTSCHOOL,ENDSCHOOL,SSLAT,SSLONG,ESLAT,ESLONG,
176         &         TIMESTART,TIMEEND,FINISHED)
177                  END IF
178     C
179                    WRITE(6,72)
180        72          FORMAT(" End of Time Window.")
181                  ENDIF
182                END IF
183              END DO
184
185              IF (FINISHED) BREAK
187              CURRENTRUN = CURRENTRUN + 1
188
189           END DO
190
191           WRITE(6,200)
192       200 FORMAT(" All files processed")
193
194        99 STOP
195           END
196     $CONTROL INIT,BOUNDS
197     C
198     C-------------------------------------------------
199           SUBROUTINE REVERSEBITS (BUFFER,LEN)
200     C-------------------------------------------------
201     C
202     C     THIS SUBROUTINE REVERSES THE ORDER OF ALL 8 BITS
203     C     IN EACH BYTE (2 BYTES/WORD) OF THE INPUT BUFFER.
204     C
205     C-------------------------------------------------
206     C
207           LOGICAL TEMP,BUFFER(4500)
208     C
209           DO 20 IWORD = 1, LEN
210           TEMP[0:1]  = BUFFER(IWORD)[7:1]
211           TEMP[1:1]  = BUFFER(IWORD)[6:1]
212           TEMP[2:1]  = BUFFER(IWORD)[5:1]
213           TEMP[3:1]  = BUFFER(IWORD)[4:1]
214           TEMP[4:1]  = BUFFER(IWORD)[3:1]
215           TEMP[5:1]  = BUFFER(IWORD)[2:1]
216           TEMP[6:1]  = BUFFER(IWORD)[1:1]
217           TEMP[7:1]  = BUFFER(IWORD)[0:1]
218           TEMP[8:1]  = BUFFER(IWORD)[15:1]
219           TEMP[9:1]  = BUFFER(IWORD)[14:1]
220           TEMP[10:1] = BUFFER(IWORD)[13:1]
221           TEMP[11:1] = BUFFER(IWORD)[12:1]
222           TEMP[12:1] = BUFFER(IWORD)[11:1]
223           TEMP[13:1] = BUFFER(IWORD)[10:1]
224           TEMP[14:1] = BUFFER(IWORD)[9:1]
225           TEMP[15:1] = BUFFER(IWORD)[8:1]
226           BUFFER(IWORD) = TEMP
227        20 CONTINUE
228           RETURN
229           END
```

```
230     $CONTROL INIT,BOUNDS
231     C
232     C-------------------------------------------------------
233           SUBROUTINE INITIALIZE (IN,TIMESTART,TIMEEND,FINISHED,
234          &           NUMIN,RUNS,CURRENTRUN,NUMRATES,RATE,
235          &           RSTART,REND,IRATE,A,B,C,D,RR,TR,PULSE,BW)
236     C-------------------------------------------------------
237     C
238     C     THIS SUBROUTINE INITIALIZES THE INPUT & OUTPUT FILES
239     C     AND PROMPTS THE USER FOR INSTRUCTIONS TO PROCESS
240     C     THE INPUT DATA.
241     C
242     C-------------------------------------------------------
243     C
244           INTEGER FNUM,YEAR,DAY,HOUR,MIN,SEC,
245          &        START,CURRENTRUN,RUNS(100),SDAY,SHOUR,SMIN,SSEC,
246          &        EDAY,EHOUR,EMIN,ESEC,SD,SH,SM,SS,ED,EH,EM,ES
247           CHARACTER DATE*27,CRUISE*4
248           LOGICAL FINISHED
249           DOUBLE PRECISION RSTART(100),REND(100)
250           REAL RATE(100)
251           DOUBLE PRECISION TIMESTART,TIMEEND,TIMEBUF
252     C
253     C
254        31 FORMAT(" Request for tape mount in progress")
255     C
256           FINISHED = .FALSE.
257     C
258        20 WRITE(6,33)
259        33 FORMAT(" Time frame for data to be processed:"/
260         &       " Processing begins with: (Day, hour, min., sec.) ")
261           ACCEPT SDAY, SHOUR, SMIN, SSEC
262           CALL COMPRESSTIME (TIMESTART,SDAY,SHOUR,SMIN,SSEC,0)
263           WRITE(6,34)
264        34 FORMAT(" Processing  ends  with: (Day, hour, min., sec.) ")
265           ACCEPT EDAY, EHOUR, EMIN, ESEC
266           CALL COMPRESSTIME (TIMEEND,EDAY,EHOUR,EMIN,ESEC,0)
267           WRITE(6,211)
268       211 FORMAT(" Enter the Cruise # (1st character alphabetic,"/
269         &       "  followed by 3 digits, eg. P026) ")
270           ACCEPT CRUISE
271        53 WRITE(6,201)
272       201 FORMAT(" Enter the number of Input runs to be processed ")
273           ACCEPT NUMIN
274           IF (NUMIN.EQ.0) GO TO 99
275           IF (NUMIN.LE.100) GO TO 51
276           WRITE(6,72)
277        72 FORMAT(" No more than 100 runs allowed on a single pass...")
278           GO TO 53
279     C
280        51 DO 100 J = 1,NUMIN
281     C
282           WRITE(6,301)
```

```
      301 FORMAT(' Enter Run #')
          READ(5,*) RUNS(J)
      100 CONTINUE
C
       58 WRITE(6,*) 'Enter the number of Rate windows '
          ACCEPT NUMRATES
          IF (NUMRATES.GT.0 .AND. NUMRATES.LE.100) GO TO 57
          WRITE(6,*) 'Unacceptable number of rate windows'
          GO TO 58
C
       57 DO 56 J = 1,NUMRATES
          WRITE(6,*) 'Enter rate (per min), start time (day,hour,min,sec),'
          WRITE(6,*)'  end time (day,hour,min,sec)'
          ACCEPT IRATE,SD,SH,SM,SS,ED,EH,EM,ES
          RATE(J) = 60.0 / IRATE
          CALL COMPRESSTIME (RSTART(J),SD,SH,SM,SS,0)
          CALL COMPRESSTIME (REND(J),ED,EH,EM,ES,0)
       56 CONTINUE
          WRITE(6,41)
       41 FORMAT(' Reciever calibration: Enter a,b,c,d, parameters    :')
          READ(5,*) A,B,C,D
          WRITE(6,36)
       36 FORMAT(' Source level: ( dB re 1 mbar at 1 m.)   :')
          READ (5,*) SL
          WRITE(6,35)
       35 FORMAT(' Recieve response: (dB ref.1volt per mubar)   :')
          READ(5,*) RR
          WRITE(6,43)
       43 FORMAT(' Transmit response: (dB ref.mubar per volt at 1m)   :')
          READ(5,*) TR
          WRITE(6,37)
       37 FORMAT(' Pulse length msec.   :')
          READ(5,*) PULSE
          WRITE(6,38)
       38 FORMAT(' Equivalent Ideal Beam   :')
          READ(5,*) BW
          IRATE = 1
C
          CURRENTRUN = 1
C
C
          WRITE(6,31)
C     GET MPE FILE NUMBER FOR TAPE
       10 IN = FNUM(15)
C
C     PRINT THE INITIAL VALUES OF THE PARAMETERS
C
          CALL DATELINE (DATE)
          WRITE(6,300) DATE
      300 FORMAT(' SCHOOLSA.PROG.ACOUSTIC',20X,A27)
          WRITE(6,101) SDAY,SHOUR,SMIN,SSEC
      101 FORMAT(' Start Time Window',I4,3I3)
          WRITE(6,102) EDAY,EHOUR,EMIN,ESEC
```

```
337      102 FORMAT(' End   Time Window',I4,3I3)
338          WRITE(6,103) CRUISE
339      103 FORMAT(' Cruise Name ',A4)
342          WRITE(6,105) NUMIN
343      105 FORMAT(' ',I3,' Runs to be processed on tape.')
344          DO 106 J=1,NUMIN
345          WRITE(6,107) RUNS(J)
346      107 FORMAT(' Run #',I3,' will be processed')
347      106 CONTINUE
348          WRITE(6,108) NUMRATES
349      108 FORMAT(' ',I3,' Rate Windows are defined')
350          DO 109 J=1,NUMRATES
351          CALL EXPANDTIME (RSTART(J),I1,I2,I3,I4,I5)
352          CALL EXPANDTIME (REND(J),I6,I7,I8,I9,I10)
353          WRITE(6,110) 60./RATE(J),I1,I2,I3,I4,I5,I6,I7,I8,I9,I10
354      110 FORMAT(' ',I3,'/min. from ',5I3,' to ',5I3)
355      109 CONTINUE
356          WRITE(6,111) A,B,C,D
357      111 FORMAT(' Reciver calibration parameters.',4F10.4)
358          WRITE(6,112) SL
359      112 FORMAT(' Source level.',F8.2)
360          WRITE(6,113) RR
361      113 FORMAT(' Recieve response.',F8.2)
362          WRITE(6,114) TR
363      114 FORMAT(' Transmit response.',F8.2)
364          PULSE = 1.5 * PULSE / 2.0
365          WRITE(6,115) PULSE
366      115 FORMAT(' Ct/2.',F8.2)
367          WRITE(6.116) BW
368      116 FORMAT(' Equivalent Ideal Beam.',F12.6)
369          RR = 10.0**(RR/20.0)
370          TR = 10.0**(SL/20.0)
371          RETURN
372      C
373         99 STOP
374          END
375      $CONTROL FILE=S-20,INIT,BOUNDS
376      C
377      C-------------------------------------------------
378           SUBROUTINE IN BUFFER (LEN,BUFFER,TIMESTART,TIMEEND,TIMETRANS,
379          &          BUFEND,NUMRATES,RATE,RSTART,REND,IRATE,TRANSRATE,
380          &          STARTBOX,ENDBOX,STARTBOTTOM,ENDBOTTOM,STARTTOP,ENDTOP,
381          &          SCHOOLTRANS,FINISHED,NEWRUN,
382          &          SPOSTIME,EPOSTIME,SPOSLAT,SPOSLONG,EPOSLAT,EPOSLONG,
383          &          STARTSCHOOL,ENDSCHOOL,SSLAT,SSLONG,ESLAT,ESLONG,
384          &          NOBOX,NEWSCHOOL,INSCHOOL,SALIST,
385          &          A,B,C,D,SL,RR,TR,PULSE,BW,SCHOOLSIGMA,SCHOOLSIGMASQR)
386      C-------------------------------------------------------------------
387      C
388      C
389           LOGICAL BUFFER(4500),FINISHED,NEWSCHOOL,NOBOX,NEWRUN,INSCHOOL
390           INTEGER RD,RH,RM,RS,RMS,START,SCHOOLTRANS
391           REAL RATE(100),TIME(1497),WIDE(1497),SALIST(4000)
392           DOUBLE PRECISION TIMEEND,TIMETRANS,RSTART(100),REND(100)
393           DOUBLE PRECISION STARTSCHOOL,ENDSCHOOL,STARTBOX,ENDBOX
394           DOUBLE PRECISION SPOSTIME,EPOSTIME,BUFEND,SCHOOLTIME,TIMESTART
395      C
396      C
397           START = 4
398           BUFTRANS = 0.0
399           BUFINTR = 0.0
400      C
401          10 CALL GETECHO (START,LEN,BUFFER,TIME,WIDE,NUMTIMES)
402      C
403      C
404           IF (TIMETRANS.GT.TIMEEND) GO TO 70
405           IF (NUMTIMES.EQ.0) GO TO 30
406      C
407           IF (TIMETRANS.LE.STARTBOX.OR.NEWRUN)
408              NEWSCHOOL = .TRUE.
```

```
409           NEWRUN = .FALSE.
410         END IF
411   C
412         IF (TIMETRANS.LT.STARTBOX.OR.TIMETRANS.LT.TIMESTART) GO TO 60
413   C
414         DO WHILE (TIMETRANS.GE.EPOSTIME)
415           SPOSLAT = EPOSLAT
416           SPOSLONG = EPOSLONG
417           SPOSTIME = EPOSTIME
418           CALL GET POSITION (TIMETRANS,SPOSLAT,SPOSLONG,
419        &                     EPOSLAT,EPOSLONG,SPOSTIME,EPOSTIME)
420         END DO
421   C
422         IF(TIMETRANS.GE.STARTBOX.AND.TIMETRANS.LT.ENDBOX)
423           INSCHOOL = .TRUE.
424           IF(NEWSCHOOL)
425   C
426   C           CLEAR SALIST ARRAY
427               DO I = 1,4000
428                 SALIST(I) = 0
429               END DO
430   C
431   C
432   C           CALCULATE START POSITION OF SCHOOL (SSLAT-SSLONG)
433   C
434               STARTSCHOOL = TIMETRANS
435               CALL DEGMI (SPOSLAT,SPOSLONG,EPOSLAT,EPOSLONG,
436        &                  COURSE,DISTANCE)
437               DIST = (STARTSCHOOL-SPOSTIME)/(EPOSTIME-SPOSTIME)*DISTANCE
438               CALL LAT LONG (SPOSLAT,SPOSLONG,COURSE,DIST,SSLAT,SSLONG)
439               NEWSCHOOL = .FALSE.
440             END IF
441   C
442             SCHOOLTRANS = SCHOOLTRANS + 1
443   C
444   C         CALCULATE TOP AND BOTTOM OF BOX FOR THIS TRANSMISSION
446   C
447             RATIO = (TIMETRANS-STARTBOX)/(ENDBOX-STARTBOX)
448             TOP = STARTTOP-(STARTTOP-ENDTOP)*RATIO
449             BOTTOM = STARTBOTTOM-(STARTBOTTOM-ENDBOTTOM)*RATIO
450   C
451             CALL ADD IN SCHOOL (TIME,WIDE,NUMTIMES,TOP,BOTTOM,
452        &                  A,B,C,D,RR,TR,PULSE,BW,TXSIGMA)
453   C
454             SALIST(SCHOOLTRANS) = TXSIGMA
455             SCHOOLSIGMA = SCHOOLSIGMA + TXSIGMA
456             SCHOOLSIGMASQR = SCHOOLSIGMASQR + TXSIGMA ** 2.0
457   C
458   C
459         ELSE
460   C
461               CALL SCHOOL END(TIMETRANS,NOBOX,NEWSCHOOL,INSCHOOL,
462        &           STARTBOX,ENDBOX,STARTBOTTOM,ENDBOTTOM,STARTTOP,ENDTOP,
463        &           SCHOOLTRANS,SCHOOLSIGMA,SCHOOLSIGMASQR,SALIST,
464        &           SPOSTIME,EPOSTIME,SPOSLAT,EPOSLAT,SPOSLONG,EPOSLONG,
465        &           STARTSCHOOL,ENDSCHOOL,SSLAT,SSLONG,ESLAT,ESLONG,
466        &           TIMESTART,TIMEEND,FINISHED)
467         ENDIF
468   C
469   C
470   C     TRANSMISSIONS OCCUR EVERY "RATE" SECONDS.
471      60 IF (TIMETRANS.LE.REND(IRATE)) GO TO 30
472   C     LOOKUP THE RATE TO BE USED AT THIS TIME
473         DO 31 J = 1,NUMRATES
474         IF (TIMETRANS.GE.RSTART(J) .AND. TIMETRANS.LE.REND(J)) GO TO 32
475      31 CONTINUE
476         WRITE(6,8) "TIME STAMP",TIMETRANS,"(SECONDS) NOT IN RATE WINDOW"
477         STOP
478      32 IRATE = J
479         TRANSRATE = RATE(IRATE)
480         CALL EXPANDTIME (TIMETRANS,RD,RH,RM,RS,RMS)
481         WRITE(6,33) 60.0/RATE(IRATE),RD,RH,RM,RS,RMS
```

```
      33 FORMAT(' New rate is ',I4,' at time ',4I3,I4)
      30 TIMETRANS = TIMETRANS + TRANSRATE
C
         IF (START-LEN+2) 10,10,99
      70 FINISHED = .TRUE.
      99 RETURN
         END
$CONTROL INIT,BOUNDS,LOCATION
C
C-------------------------------------------------------
         SUBROUTINE GETECHO (START,LEN,BUFFER,TIME,WIDE,NUMTIMES)
C-------------------------------------------------------
C
C     THIS SUBROUTINE CONVERTS THE BINARY CODED VALUES FOR TIME AND
C     VOLTAGE TO METERS AND MILLIVOLTS
C
C-------------------------------------------------------
C
         INTEGER START
         REAL TIME(1497),WIDE(1497)
         LOGICAL BUFFER(4500),TEMP1
C
         II = 0
         DO 20 J = START, LEN-2, 3
         IF (II.GE.1497) GO TO 25
         II = II + 1
         TIME(II) = INT(BUFFER(J)[4:12])
         TEMP1 = .NOT. BUFFER(J+1)
         WIDE(II)  = INT(TEMP1[4:12])
C     CONVERT A/D VALUE TO VOLTS
         WIDE(II) = WIDE(II)/409.6
C     CONVERT TIME IN 10THS OF MILLISECONDS TO DEPTH IN METERS
         TIME(II) = TIME(II) * 0.075
         IF (II-1) 10,10,30
      30 IF (TIME(II)-TIMEOLD) 40,40,10
      10 CONTINUE
         TIMEOLD = TIME(II)
      20 CONTINUE
         NUMTIMES = II
         GO TO 60
      25 WRITE(6,26)
      26 FORMAT(' WARNING: BUFFER OVERFLOW. ONLY 1497 COUNTS SAVED')
      40 NUMTIMES = II - 1
      60 START = J
         RETURN
         END
```

```
$CONTROL INIT,BOUNDS
C
C-----------------------------------------------------------------
      SUBROUTINE COMPRESSTIME (TIME,DAY,HOUR,MIN,SEC,MSEC)
C-----------------------------------------------------------------
C
C     FUNCTION:  COMPRESSES THE DATE AND TIME INTO DECIMAL SECONDS
C
C     CALLED BY: GETNEXTRUN
C
C     CALLS: ((NIL))
C
C     PARAMETER DEFINITIONS
C
C     TIME - THE TIME IN DECIMAL SECONDS.
C     DAY - INTEGER
C     HOUR - "
C     MIN  - "
C     SEC  - "
C     MSEC - "
C
C     MODIFIED PARAMETERS - TIME
C
C     PARAMETER DECLARATIONS
C
      DOUBLE PRECISION TIME
      INTEGER DAY,HOUR,MIN,SEC,MSEC
C
C     LOCAL VARIABLES  - ((NIL))
C
C
      IF (DAY.LT.0  .OR. DAY.GT.366) GO TO 10
      IF (HOUR.LT.0 .OR. HOUR.GT.24) GO TO 10
      IF (MIN.LT.0  .OR. MIN.GT.60) GO TO 10
      IF (SEC.LT.0  .OR. SEC.GT.60) GO TO 10
      IF (MSEC.GE.0 .AND.MSEC.LE.999) GO TO 20
C
   10 WRITE(6,*) "((COMPRESSTIME)) Error in time parameter"
      WRITE(6,*) DAY,HOUR,MIN,SEC,MSEC
C
   20 TIME = DAY*86400.
      TIME = TIME + HOUR*3600.
      TIME = TIME + MIN*60.
      TIME = TIME + SEC
      TIME = TIME + MSEC/1000.
      RETURN
      END
$CONTROL INIT,BOUNDS
C
C-----------------------------------------------------------------
      SUBROUTINE EXPANDTIME (TIME,DAY,HOUR,MIN,SEC,MSEC)
C-----------------------------------------------------------------
      DOUBLE PRECISION TIME,T,D,H,M,S,MS
      INTEGER DAY,HOUR,MIN,SEC,MSEC
C
      T = TIME
      D = T/86400.0
      DAY = IDINT(D)
      T = (D -DAY) *86400.0
      H = T/3600.0
      HOUR = IDINT(H)
      T = (H -HOUR) *3600.0
      M = T/60.0
      MIN = IDINT(M)
      S = (M -MIN) *60.0
      SEC = IDINT(S)
      MS = (S -SEC) *1000.0
      MSEC = IDINT(MS +.5)
      IF (1000-MSEC) 50,30,40
   50 WRITE(6,*) " ((EXPANDTIME)) Error in time parameter."
   30 SEC = SEC + 1
      MSEC = 0
   40 CONTINUE
      RETURN
      END
```

```
C-------------------------------------------------------
      SUBROUTINE GETBUFFER (BUFFER,LEN,EOF,RATE,NUMTRANS1,TIMESTART,
     &                      NUMRATES,RSTART,REND,IRATE,TIMEX,
     &                      TRANSRATE,IN,TIME1,TIME2)
C-------------------------------------------------------
C     GETS A BUFFER FROM THE INPUT TAPE, AND CALCULATES THE
C     ACTUAL TRANSMISSION RATE OBSERVED BETWEEN TWO BUFFERS.
C
C     SUBROUTINE PARAMETERS
      LOGICAL BUFFER(4500),BUFFER2(4500),EOF,RATECHANGED,MISREAD
      REAL RATE(100)
      DOUBLE PRECISION RSTART(100),REND(100),TIMEX,TIME1,TIME2,TIMESTART
      INTEGER BUFFERCONTROL(4502),LEN
      INTEGER LEN2,CONTROL,YEAR1,DAY1,HOUR1,SEC1,YEAR2,DAY2,HOUR2,SEC2
      COMMON /BUFFERCONTROL/BUFFERCONTROL
      EQUIVALENCE (BUFFERCONTROL(1),CONTROL),(BUFFERCONTROL(2),LEN2),
     &           (BUFFERCONTROL(3),BUFFER2)
      SYSTEM INTRINSIC FREAD,PRINTFILEINFO
C
C     ON FIRST READ, BOTH THE 1ST BUFFER AND THE 2ND BUFFER MUST
C     BE READ TO CALCULATE THE TRUE RATE BETWEEN SUCCESSIVE TIME
C     STAMPS, AND TO FIND START TIME. READ FIRST BUFFER.
      IF (CONTROL.EQ.0) THEN
    1    LEN = FREAD (IN,BUFFER,4500)
         IF (.CC.) 10,11,12
   10       CALL PRINTFILEINFO (IN)
            GO TO 1
   12       EOF = .TRUE.
            RETURN
C     FIRST BUFFER READ SUCCESSFULY
   11    CALL CHECKLEN (BUFFER,LEN)
         CALL REVERSEBITS (BUFFER,LEN)
      ENDIF

C     IF CONTROL IS 2, THEN
C     AS EOF WAS DETECTED ON THE PREVIOUS ATTEMPT TO GET THE 2ND
C     BUFFER, THEN A TRUE EOF OF FILE SHOULD BE RETURNED NOW.
      IF (CONTROL.EQ.2) THEN
         EOF = .TRUE.
         CONTROL = 0
         RETURN
      ENDIF

C     ON NEXT AND SUBSEQUENT (NON EOF) READS, THE 2ND BUFFER IS
C     TRANSFERRED TO THE FIRST BUFFER, BEFORE THE ANOTHER BUFFER
C     IS ACQUIRED FROM TAPE.
         IF (CONTROL.EQ.1) THEN
         LEN = LEN2
         DO J = 1,LEN
            BUFFER(J) = BUFFER2(J)
         END DO
         ENDIF

```

```
      IF(CONTROL.EQ.0 .OR. CONTROL.EQ.1) THEN
  100     CALL GETDATE (BUFFER,YEAR1,DAY1,HOUR1,MINT1,SEC1)
          CALL COMPRESSTIME (TIME1,DAY1,HOUR1,MINT1,SEC1,0)
          CALL COUNTTRANS (BUFFER,NUMTRANS1,LEN)
C     READ SECOND BUFFER
        MISREAD = .FALSE.
    2    LEN2 = FREAD (IN,BUFFER2,4500)
        IF (.CC.) 13,15,16
   13   CALL PRINTFILEINFO(IN)
        MISREAD = .TRUE.
        GO TO 2
C
C       FOUND EOF ON INPUT TAPE
C       USE LAST CALCULATED RATE, OR WHEN ONLY 1 BUFFER IN THE FILE,
C       THE DEFAULT MUST BE USED...  RATE(IRATE)
   16   CONTROL = 2
      TRANSRATE = RATE(IRATE)
        RETURN

C       2ND BUFFER READ SUCCESSFULLY
   15   CALL CHECKLEN (BUFFER2,LEN2)
        CALL REVERSEBITS (BUFFER2,LEN2)
        CALL GETDATE (BUFFER2,YEAR2,DAY2,HOUR2,MIN2,SEC2)
        CALL COMPRESSTIME (TIME2,DAY2,HOUR2,MIN2,SEC2,0)
      IF (TIME2.LE.TIMEX.OR.TIME2.LE.TIMESTART) THEN
C     ON NEXT AND SUBSEQUENT (NON EOF) READS, THE 2ND BUFFER IS
C     TRANSFERRED TO THE FIRST BUFFER, BEFORE THE ANOTHER BUFFER
C     IS ACQUIRED FROM TAPE.
        LEN = LEN2
        DO J = 1,LEN
          BUFFER(J) = BUFFER2(J)
        END DO
      GO TO 100


      ELSE
        CALL CHECKTIME (TIME1,TIME2,RATE,RSTART,REND,NUMRATES,IRATE,
     &                  RATECHANGED)

C       IF A RATE CHANGE OCCURS BETWEEN THE BUFFER START TIMES

        IF (RATECHANGED.OR.MISREAD) THEN
C         THE DEFAULT RATE MUST BE USED WHEN A CALCULATED RATE IS
            TRANSRATE = RATE(IRATE)
            WRITE(6,124) 60./RATE(IRATE)
  124       FORMAT(" Pulse rate change from:",I4)
        ELSE
C         AS NO RATE CHANGE OCCURS, CALCULATE THE TRUE RATE.
          TRANSRATE = (TIME2-TIME1) / NUMTRANS1
        ENDIF
        CONTROL = 1
      ENDIF
      ELSE

        WRITE(6,8) "(GETBUFFER) CONTROL PARAMETER ERROR."
      ENDIF
      RETURN
      END
```

```
C
C-----------------------------------------------------------------------
      SUBROUTINE COUNTTRANS (BUFFER,NUMTRANS,LENGTH)
C-----------------------------------------------------------------------
C     COUNTS THE NUMBER OF TRANSMISSIONS IN AN INPUT BUFFER.
C
C     SUBROUTINE PARAMETERS
      LOGICAL BUFFER(4500)
      INTEGER LENGTH,NUMTRANS
C
C     LOCAL VARIABLES
      LOGICAL COMPARABLE
      INTEGER TIME1,TIME2
C
      NUMTRANS = 1
      COMPARABLE = .FALSE.
      DO J = 4, LENGTH-2, 3
         TIME2 = INT(BUFFER(J)[4:12])
         IF (COMPARABLE) THEN
            IF (TIME2.LE.TIME1) NUMTRANS = NUMTRANS + 1
         ELSE
            COMPARABLE = .TRUE.
         ENDIF
         TIME1 = TIME2
      END DO
      RETURN
      END
C-----------------------------------------------------------------------
      SUBROUTINE CHECKLEN (BUFFER,LEN)
C-----------------------------------------------------------------------
C     THIS SUBROUTINE CHECKS THE LENGTH OF INPUT BUFFERS.
C
C     SUBROUTINE PARAMETERS
      LOGICAL BUFFER(4500)
      INTEGER LEN
C
C     LOCAL VARIABLES
      REAL XTRANS
C
C     CHECK FOR A VALID BUFFER LENGTH
   40 XTRANS = (LEN-3.0) / 3.0
      IF (XTRANS.NE.AINT(XTRANS)) THEN
         WRITE(6,42) LEN
   42    FORMAT(' WARNING: INPUT BUFFER length of',
     &          I5,' inappropriate.')
      ENDIF
      RETURN
      END
C-----------------------------------------------------------------------
      SUBROUTINE CHECKTIME (TIME1,TIME2,RATE,RSTART,REND,NUMRATES,
     &                      IRATE,RATECHANGED)
C-----------------------------------------------------------------------
C     THIS SUBROUTINE CHECKS IF A RATE CHANGE OCCURS WITHIN A TIME
C     WINDOW.
C
C     SUBROUTINE PARAMETERS
      DOUBLE PRECISION TIME1,TIME2,RSTART(100),REND(100)
      REAL RATE
      LOGICAL RATECHANGED
      INTEGER NUMRATES,IRATE
C
C     LOCAL VARIABLES
C
      RATECHANGED = .FALSE.
      DO J = 1, NUMRATES
         IF (REND(J).GT.TIME1 .AND. REND(J).LT.TIME2) THEN
            RATECHANGED = .TRUE.
            IRATE = J
            BREAK
         ENDIF
      END DO
      RETURN
      END
```

```
C
C
C
C-----------------------------------------------------------------
      SUBROUTINE TRACK BOXES (TIMESTART,NOBOX,STARTBOX,ENDBOX,
     &             STARTBOTTOM,ENDBOTTOM,STARTTOP,ENDTOP)
C-----------------------------------------------------------------
C
C
C
C     FUNCTION: LOCATES BOXES TO BE PROCESSED IN FILE FTN11
C               AND RETURNS BOX PARAMETERS.
C
C
      LOGICAL NOBOX
      INTEGER SBOT,EBOT,SSTOP,ETOP
      DOUBLE PRECISION TIMESTART,STARTBOX,ENDBOX
C
   61 READ(11,12,ERR=62,END=63) STARTBOX,ENDBOX,SBOT,EBOT,SSTOP,ETOP
   12 FORMAT(2F14.4,4F10.3)
C
      IF(TIMESTART.GT.ENDBOX)
        GO TO 61
      ELSE
        GO TO 65
      END IF
C
   62 WRITE(6,8) ' READ ERROR IN BOXES FILE'
      STOP
   63 WRITE(6,8) ' End boxes file.'
      NOBOX = .TRUE.
      RETURN
C
   65 CONTINUE
      STARTBOTTOM = SBOT/100.0
      ENDBOTTOM = EBOT/100.0
      STARTTOP = SSTOP/100.0
      ENDTOP = ETOP/100.0
C
      RETURN
      END
```

```
C
C
C-------------------------------------------------------
      SUBROUTINE ADD IN SCHOOL (TIME,WIDE,NUMTIMES,TOP,BOTTOM,
     &          A,B,C,D,RR,TR,PULSE,BW,TXSIGMA)
C-------------------------------------------------------
C
C
C
C     FUCTION: FOR EACH INTERRUPT, CALCULATES INPUT VOLTAGE USING
C     CALIBRATION REGRESSION AND ADDS TVG FUNCTION. FOR EACH
C     TRANSMISSION,CALCULATES AREA SCATTERING COEFFICIENT.
C
      DIMENSION TIME(1497),WIDE(1497)
C
      TXSIGMA = 0.0
C
      DO 100 J =1,NUMTIMES
         IF(TIME(J).GE.TOP.AND.TIME(J).LT.BOTTOM)
           DEPTH = TIME(J)
           VOLTS = WIDE(J)
C
C          CALCULATE INPUT VOLTAGE FROM CALIBRATION REGRESSION
C
C
           VOLTS = 10.0**(A/20.0)*VOLTS**B*(DEPTH/3.0)**(C/20.0)
           VOLTS = VOLTS*10.0**(D*(2.0*DEPTH-6.0)/20.0)
C
C          CALCULATE RECIEVED PRESSURE
           RP = VOLTS/RR
C          CALCULATE RECIEVED INTENSITY
           RI = 6.667E-9 * RP**2.0
C          CALCULATE TRANSMITTED INTENSITY
           TI = 6.667E-9 * TR**2.0
C          CALCULATE VOLUME SCATTERING COEFFICIENT
           SV = RI*DEPTH**2.0*2.718282**(.00562*DEPTH)
           SV = SV/TI/PULSE/BW
C
C          CONVERT TO AREA SCATTERING COEFFICIENT
           SA = SV * .075
C
C
C          SUM AREA SCATTERING COEFFICIENTS FOR WHOLE WATER COLUMN
C
           TXSIGMA = TXSIGMA + SA
C
         END IF
  100 CONTINUE
      RETURN
      END
```

```
C
C
C
C-------------------------------------------------------------
      SUBROUTINE DEGMI (SLAT,SLONG,ELAT,ELONG,COURSE,DISTANCE)
C-------------------------------------------------------------
C
C
C
C     FUNCTION: CALCULATES COURSE AND DISTANCE BETWEEN
C     TWO LAT.LONG. POSITIONS
C
      LOGICAL NORTH,EAST,SOUTH,WEST
C
      NORTH = .FALSE.
      EAST = .FALSE.
      SOUTH = .FALSE.
      WEST = .FALSE.
      R1 = ALOG(TAN(.785398+.008727*SLAT))
      R2 = ALOG(TAN(.785398+.008727*ELAT))
      Y = 3.141593*(SLONG-ELONG)/180.0
      X = R2-R1
      IF (ABS(X).LT..000001) GO TO 20
      IF (SLAT-ELAT) 10,20,30
   10 NORTH = .TRUE.
      GO TO 20
   30 SOUTH = .TRUE.
   20 IF (SLONG-ELONG) 40,50,60
   40 WEST = .TRUE.
      GO TO 70
   60 EAST = .TRUE.
      GO TO 70
   50 IF (NORTH) COURSE = 360.0
      IF (SOUTH) COURSE = 180.0
   70 IF (NORTH.OR.SOUTH) GO TO 80
      DISTANCE = 60.0*(SLONG-ELONG)*COS(.017453*SLAT)
      IF (WEST) COURSE = 270.0
      IF (EAST) COURSE = 90.0
      GO TO 100
   80 C = ATAN(Y/X)
      DISTANCE = 60.0*(ELAT-SLAT)/COS(C)
      C = 360.0 * C /6.28319
      IF (NORTH.AND.EAST) COURSE = C
      IF (SOUTH) COURSE = 180.0 + C
      IF (WEST.AND.NORTH) COURSE = 360.0 + C
  100 DISTANCE = ABS(DISTANCE)
      RETURN
      END
C
C
C
C
C-------------------------------------------------------------
      SUBROUTINE LAT LONG (SLAT,SLONG,COURSE,DISTANCE,LAT,LONG)
C-------------------------------------------------------------
C
C
C     FUNCTION: CALCULATES THE LAT/LONG OF A POSITION THAT IS
C     (COURSE) DEG. AND (DISTANCE) MI. FROM A KNOWN POSITION SLAT/SLONG
C
      REAL LAT,LONG
      IF(COURSE-90.0) 20,10,20
   20 IF(COURSE-270.0) 30,10,30
   10 LONG = SLONG-DISTANCE/(60.0*COS(.017453*SLAT))
      LAT = SLAT
      GO TO 40
   30 LAT = SLAT+DISTANCE*COS(.017453*COURSE)/60.0
      R1 = ALOG(TAN(.785398+.008727*SLAT))
      R2 = ALOG(TAN(.785398+.008727*LAT))
      X = R2-R1
      R3 = X/COS(.017453*COURSE)
      Y = R3*SIN(.017453*COURSE)
      LONG = SLONG-180.0*Y/3.141593
   40 CONTINUE
      RETURN
      END
```

```
962   C
963   C-------------------------------------------------------
964         SUBROUTINE GET POSITION (TIME,SLAT,SLONG,ELAT,ELONG,
965        &                         SPOSTIME,EPOSTIME)
967   C-------------------------------------------------------
968   C
969   C
970   C
971   C     FUNCTION: GETS POSITION PARAMETERS FROM FILE FTN12
972   C
973   C
974         INTEGER DAY,HOUR,MIN,SEC
975         DOUBLE PRECISION TIME,SPOSTIME,EPOSTIME
976   C
977      30 READ(12,13,END=40) DAY,HOUR,MIN,SEC,LATD,LATM,LATS,
978        &                   LONGD,LONGM,LONGS
979      13 FORMAT(10I3)
980   C
981         CALL COMPRESSTIME (EPOSTIME,DAY,HOUR,MIN,SEC,0)
982   C
983   C     CONVERT LAT./LONG. TO DECIMAL DEGREES
984   C
985         ELAT = LATD + (LATM + LATS/60.0)/60.0
986         ELONG = LONGD + (LONGM + LONGS/60.0)/60.0
987   C
988      20 IF (TIME.GE.EPOSTIME)
989            SLAT = ELAT
990            SLONG = ELONG
991            SPOSTIME = EPOSTIME
992            GO TO 30
993         END IF
994   C
995         IF (SPOSTIME.EQ.0.0)
996            WRITE(6,*) " CAN NOT PROCESS DATA BEFORE FIRST POSITION"
997            STOP
998         END IF
999         RETURN
1000     40 WRITE(6,*) " End of position file."
1001        RETURN
1002        END
1003  C
1004  C
1005  C
1006  $CONTROL INIT,BOUNDS
1007  C
1008  C-------------------------------------------------------
1009        SUBROUTINE GETDATE (BUFFER,YEAR,DAY,HOUR,MIN,SEC)
1010  C-------------------------------------------------------
1011  C
1012  C     THIS SUBROUTINE CONVERTS THE DATE AND TIME INFORMATION
1013  C     IN THE 1ST 6 BYTES OF A BUFFER INTO INTEGER VALUES.
1014  C
1015  C-------------------------------------------------------
1016  C
1017        INTEGER YEAR,DAY,HOUR,MIN,SEC
1018        LOGICAL BUFFER(4500)
1019  C
1020        YEAR = INT(BUFFER(3)[8:4])*10 + INT(BUFFER(3)[12:4])
1021        DAY  = INT(BUFFER(3)[4:4])*100 + INT(BUFFER(2)[8:4])*10
1022       &       + INT(BUFFER(2)[12:4])
1023        HOUR = INT(BUFFER(2)[0:4])*10 + INT(BUFFER(2)[4:4])
1024        MIN  = INT(BUFFER(1)[8:4])*10 + INT(BUFFER(1)[12:4])
1025        SEC  = INT(BUFFER(1)[0:4])*10 + INT(BUFFER(1)[4:4])
1027        RETURN
1028        END
```

```
1029     $CONTROL INIT,BOUNDS
1030     C
1031     C
1032     C-----------------------------------------------------------
1033           SUBROUTINE SCHOOL END(TIMETRANS,NOBOX,NEWSCHOOL,INSCHOOL,
1034          &          STARTBOX,ENDBOX,STARTBOTTOM,ENDBOTTOM,STARTTOP,ENDTOP,
1035          &          SCHOOLTRANS,SCHOOLSIGMA,SCHOOLSIGMASQR,SALIST,
1036          &          SPOSTIME,EPOSTIME,SPOSLAT,EPOSLAT,SPOSLONG,EPOSLONG,
1037          &          STARTSCHOOL,ENDSCHOOL,SSLAT,SSLONG,ESLAT,ESLONG,
1038          &          TIMESTART,TIMEEND,FINISHED)
1039     C-----------------------------------------------------------------
1040     C
1041     C
1042     C
1043     C     FUNCTION: CALCULATES SCHOOL END TIME AND POSITION, AVERAGE AREA-
1044     C     SCATTERING AND STANDARD DEVIATION,FREQUENCY DISTRIBUTION OF
1045     C     SCATTERING LEVELS, AND WRITES OUTPUT TO FILE FTN19 & FTN06
1046     C
1047     C
1048           INTEGER SCHOOLTRANS,BIN(16),LENGTH
1049           INTEGER SDAY,SHOUR,SMIN,SSEC,SMSEC,EDAY,EHOUR,EMIN,ESEC,EMSEC
1050           INTEGER SLATD,SLATM,SLATS,SLONGD,SLONGM,SLONGS
1051           INTEGER ELATD,ELATM,ELATS,ELONGD,ELONGM,ELONGS
1053           LOGICAL NEWSCHOOL,NOBOX,INSCHOOL,FINISHED
1054           DOUBLE PRECISION TIMETRANS,STARTBOX,ENDBOX,STARTSCHOOL,ENDSCHOOL,
1055          &                 SPOSTIME,EPOSTIME,SCHOOLTIME,TIMESTART,TIMEEND
1056           REAL LOBIN,HIBIN,SALIST(4000)
1057           CHARACTER STRING*100,BLANKS*100
1058           DATA BLANKS/' '/
1059     C
1060           ENDSCHOOL = ENDBOX
1061           CALL TRACK BOXES (TIMETRANS,NOBOX,STARTBOX,ENDBOX,
1062          &                  STARTBOTTOM,ENDBOTTOM,STARTTOP,ENDTOP)
1063           IF (STARTBOX.GT.ENDSCHOOL.OR.NOBOX.OR.FINISHED)
1064              INSCHOOL = .FALSE.
1065     C
1066              IF (SCHOOLTRANS.LE.1) RETURN
1067              IF (SCHOOLSIGMA.EQ.0.0) RETURN
1068     C
1069     C        CALCULATE END POSITION OF SCHOOL (ESLAT-ESLONG)
1070     C
1071           IF (TIMEEND.LT.ENDSCHOOL) ENDSCHOOL = TIMEEND
1072     C
1073              CALL DEGNI (SPOSLAT,SPOSLONG,EPOSLAT,EPOSLONG,
1074          &               COURSE,DISTANCE)
1075              DIST = (ENDSCHOOL-SPOSTIME)/(EPOSTIME-SPOSTIME)*DISTANCE
1076              CALL LAT LONG (SPOSLAT,SPOSLONG,COURSE,DIST,ESLAT,ESLONG)
1077     C
1078     C        CALCULATE TIME OF MIDDLE OF SCHOOL
1079     C
1080              SCHOOLTIME = STARTSCHOOL + (ENDSCHOOL-STARTSCHOOL)/2.0
1081     C
1082              AVGSA = SCHOOLSIGMA/SCHOOLTRANS
```

```
1083            SDEV = SCHOOLSIGMASQR-SCHOOLSIGMA**2.0/SCHOOLTRANS
1084            SDEV = (SDEV/(SCHOOLTRANS-1)) ** 0.5
1085     C
1086     C
1087     C       OUTPUT SCHOOL PARAMETERS AND HISTOGRAM TO NEW PAGE
1088     C
1089            CALL EXPAND TIME (STARTSCHOOL,SDAY,SHOUR,SMIN,SSEC,SMSEC)
1090            CALL EXPAND TIME (ENDSCHOOL,EDAY,EHOUR,EMIN,ESEC,EMSEC)
1091     C
1092            WRITE(19,26)SDAY,SHOUR,SMIN,SSEC,SMSEC,EDAY,EHOUR,EMIN,ESEC,
1093           &          EMSEC,SSLAT,SSLONG,ESLAT,ESLONG,AVGSA,SDEV,
1094           &            SCHOOLTRANS
1095          26   FORMAT(10I4,4F10.5,2E12.6,I6)
1096     C
1097            WRITE(6,61) SDAY,SHOUR,SMIN,SSEC
1098          61   FORMAT('1Start time',I21,3I4)
1099            WRITE(6,62) EDAY,EHOUR,EMIN,ESEC
1100          62   FORMAT(' End time',2X,I21,3I4)
1101     C
1102            CALL DMS (SSLAT,SLATD,SLATM,SLATS)
1103            CALL DMS (SSLONG,SLONGD,SLONGM,SLONGS)
1104            CALL DMS (ESLAT,ELATD,ELATM,ELATS)
1105            CALL DMS (ESLONG,ELONGD,ELONGM,ELONGS)
1106            WRITE(6,63) SLATD,SLATM,SLATS,SLONGD,SLONGM,SLONGS
1107          63   FORMAT(' Start Lat/Long (d.m.s.)',I8,2I3,I9,2I3)
1108            WRITE(6,64) ELATD,ELATM,ELATS,ELONGD,ELONGM,ELONGS
1109          64   FORMAT(' End Lat/Long (d.m.s.)',2X,I8,2I3,I8,2I3)
1110            CALL DEGMI (SSLAT,SSLONG,ESLAT,ESLONG,COURSE,DISTANCE)
1111            WRITE(6,66) DISTANCE
1112          66   FORMAT(' School length (miles)',F15.3)
1113            WRITE(6,70) COURSE
1114          70   FORMAT(' Transect course (deg.)',F12.1)
1115            WRITE(6,65) SCHOOLTRANS
1116          65   FORMAT(' Number of transmissions',I9)
1117            WRITE(6,72) SCHOOLSIGMA
1118          72   FORMAT(' Sum of area scattering',E20.4)
1119            WRITE(6,73) SCHOOLSIGMASQR
1120          73   FORMAT(' Sum of area scattering sqrd.',E14.4)
1121            WRITE(6,67) AVGSA
1122          67   FORMAT(' Average area scattering',E18.4)
1123            WRITE(6,68) SDEV
1124          68   FORMAT(' Standard deviation',E23.4)
1125            WRITE(6,69) SDEV/AVGSA
1126          69   FORMAT(' Coefficient of variation',F12.3,6/)
1127     C
1128     C
1129            WRITE(6,59)
1130          59   FORMAT(10X,'Standard',20X,'Relative %')
1131            WRITE(6,58)
1132          58   FORMAT(10X,'Deviations',1X,'0----+----10---+----20',
1133           &  '---+----30---+----40---+----50')
1134     C
1135     C       CLEAR BIN ARRAY
```

```
1136    C
1137            DO I = 1,16
1138               BIN(I) = 0
1139            END DO
1140    C
1141    C       ACCUMULATE SCATTERING IN HALF STANDARD DEVIATION GROUPS
1142    C
1143            DO I = 1,SCHOOLTRANS
1144               LOBIN = AVGSA - 4. * SDEV
1145               HIBIN = LOBIN + .5 * SDEV
1146               DO J = 1,16
1147                  IF(SALIST(I).GT.LOBIN.AND.SALIST(I).LE.HIBIN)
1148                     BIN(J) = BIN(J) + 1
1149                     BREAK
1150                  ELSE
1151                     LOBIN = HIBIN
1152                     HIBIN = LOBIN + .5 * SDEV
1153                  END IF
1154               END DO
1155            END DO
1156    C
1157            COUNT = -3.5
1158            DO I = 1,16
1159               STRING = BLANKS
1160               LENGTH = INT(BIN(I) * 100. / SCHOOLTRANS + .5)
1161               IF (LENGTH.GE.1)
1162                  DO J = 1,LENGTH
1163                     STRING[J:1] = "*"
1164                  END DO
1165               END IF
1166    C
1167               WRITE(6,71) COUNT,STRING
1168       71      FORMAT(F19.1,2X,"I",S)
1169               COUNT = COUNT + .5
1170            END DO
1171    C
1172            NEWSCHOOL = .TRUE.
1173            SCHOOLTRANS = 0
1174            SCHOOLSIGMA = 0.0
1175            SCHOOLSIGMASQR = 0.0
1176            IF (NOBOX)
1177               WRITE(6,*) " All boxes processed"
1178               STOP
1179            END IF
1180         ELSE
1181            ENDSCHOOL = ENDBOX
1182         END IF
1183         RETURN
1184         END
1185    C
1186    C
1187    C-------------------------------------------
1188         SUBROUTINE DMS (DEC,D,M,S)
1189    C-------------------------------------------
1190    C
1191    C    FUNCTION: CONVERTS LAT/LONG FROM DECIMAL TO D.M.S FORMAT
1192    C
1193         REAL DEC
1194         INTEGER D,M,S
1195         D = INT(DEC)
1197         T = (DEC-D) * 60.0
1198         M = INT(T)
1199         S = INT((T-M) * 60.0 + .5)
1200         RETURN
1201         END
1202    C
1203    C
```

APPENDIX F

Output processing software

```
$CONTROL FILE=12,NOLIST,USLINIT,INIT
      PROGRAM RECNUMBER
C-------------------------------------------------------------------
C
C     THIS TEXT IS STORED AS RECNUM.SOURCE
C
C     A PROGRAM TO ADD RECORD NUMBERS TO BITS 111-116 OF THE RECORDS IN
C     THE OUTPUT FILE OF THE ACOUSTIC DATA PROCESSING PROGRAM
C
C-------------------------------------------------------------------
      REAL SLAT,SLONG,ELAT,ELONG,SCATT,SDEV
      INTEGER SD,SH,SM,SS,SMS,ED,EH,EM,ES,EMS
      INTEGER RECNUM,NUMTX,START,FINISH
      WRITE(6,8) "Enter record number of first record in set"
      ACCEPT START
      WRITE(6,8) "Enter record number of last record in set"
      ACCEPT FINISH
      DO 100 I=START,FINISH
      READ(12,13) SD,SH,SM,SS,SMS,ED,EH,EM,ES,EMS,SLAT,SLONG,ELAT,ELONG,
     &          SCATT,SDEV,NUMTX
   13 FORMAT(10I4,4F10.5,2E12.6,I6)
      BACKSPACE 12
      RECNUM = I
      WRITE(12,16) SD,SH,SM,SS,SMS,ED,EH,EM,ES,EMS,SLAT,SLONG,
     &            ELAT,ELONG,SCATT,SDEV,NUMTX,RECNUM
   16 FORMAT(10I4,4F10.5,2E12.6,I6,I6)
  100 CONTINUE
      STOP
      END
```

```
$CONTROL FILE=15,INIT,USLINIT,NOLIST,NOSOURCE
      PROGRAM SCATTER AVG
C
C
C-------------------------------------------------------------
C
C      THIS TEXT IS STORED AS AVGSCATT.SOURCE
C
C     A PROGRAM TO CALCULATE AVERAGE AREA SCATTERING COEFFICIENTS ,
C     STANDARD ERRORS AND TRANSECT LENGTHS FOR A SPECIFIED NUMBER OF
C     RECORDS OF THE OUTPUT FILE OF THE ACOUSTIC PROCESSING PROGRAM
C
C-------------------------------------------------------------
C
      REAL SCATT,SX,SXX,SXD,AVG,STERR,DISTANCE,SDIST,AVDIST,SDEV
      INTEGER START(100),FINISH(100),NUMDIST,NUMTX
      WRITE(6,8)" Enter number of record sets to read:   "
      ACCEPT NUM
      DO 200 I=1,NUM
         WRITE(6,8)" Enter first & last record of set",I,":  "
         ACCEPT START(I),FINISH(I)
  200 CONTINUE
      SX = 0.0
      SXX = 0.0
      SXD = 0.0
      SDIST = 0.0
      NUMDIST = 0
      DO 300 I=1,NUM
      DO 100 J = START(I),FINISH(I)
         READ(150J,20) SLAT,SLONG,ELAT,ELONG,SCATT,SDEV,NUMTX
   20    FORMAT(T41,4F10.5,2E12.6,I6)
         CALL DEGMI(SLAT,SLONG,ELAT,ELONG,COURSE,DISTANCE)
         SX = SX + SCATT
         SXX = SXX + (SDEV**2.0/NUMTX)**0.5 * DISTANCE
         NUMDIST = NUMDIST + 1
         SXD = SXD + SCATT * DISTANCE
         SDIST = SDIST + DISTANCE
  100 CONTINUE
  300 CONTINUE
      AVG = SXD/SDIST
      STERR = SXX/SDIST
      AVDIST = SDIST/NUMDIST
```

```
      WRITE(6,21) AVG,STERR
   21 FORMAT(' Avg. area scatt. coeff. and std. err.  =  ',2E12.4)
      WRITE(6,22) NUMDIST,AVDIST
   22 FORMAT(' No. of schools and avg. length  =  'I6,F8.4)
      STOP
      END

C
C
C
C-------------------------------------------------------
      SUBROUTINE DEGMI (SLAT,SLONG,ELAT,ELONG,COURSE,DISTANCE)
C-------------------------------------------------------
C
C
C     FUNCTION: CALCULATES COURSE AND DISTANCE BETWEEN
C     TWO LAT.LONG. POSITIONS
C
      LOGICAL NORTH,EAST,SOUTH,WEST
C
      NORTH = .FALSE.
      EAST = .FALSE.
      SOUTH = .FALSE.
      WEST = .FALSE.
      R1 = ALOG(TAN(.785398+.008727*SLAT))
      R2 = ALOG(TAN(.785398+.008727*ELAT))
      Y = 3.141593*(SLONG-ELONG)/180.0
      X = R2-R1
      IF (X) 30,20,10
   10 NORTH = .TRUE.
      GO TO 20
   30 SOUTH = .TRUE.
   20 IF (SLONG-ELONG) 40,50,60
   40 WEST = .TRUE.
      GO TO 70
   60 EAST = .TRUE.
      GO TO 70
   50 IF(X) 91,90,91
   90 COURSE = 0.0
      DISTANCE = 0.0
      RETURN
   91 IF (NORTH) COURSE = 360.0
      IF (SOUTH) COURSE = 180.0
   70 IF (NORTH.OR.SOUTH) GO TO 80
      DISTANCE = 60.0*(SLONG-ELONG)*COS(.017453*SLAT)
      IF (WEST) COURSE = 270.0
      IF (EAST) COURSE = 90.0
      GO TO 100
   80 C = ATAN(Y/X)
      DISTANCE = 60.0*(ELAT-SLAT)/COS(C)
      C = 360.0 * C /6.28319
      IF (NORTH.AND.EAST) COURSE = C
      IF (SOUTH) COURSE = 180.0 + C
      IF (WEST.AND.NORTH) COURSE = 360.0 + C
  100 DISTANCE = ABS(DISTANCE)
      RETURN
      END
```

APPENDIX G

Echo plotting software

```
!JOB ECOPLOT,USER.ACOUSTIC:OUTCLASS=SP12:TIME=500:INPRI=7
!CONTINUE
!ECOPLOT P08SW07.12.SP12,$NULL
Y           ((PRINT BUF. SIZE))
Y           ((PRINT DATES))
N           ((PRINT ECHOS))
Y           ((PLOT ECHOS))
10,20       ((LAYER TOP,BOTTOM))
500         ((ECHO SIZE IN MV. PER 1/2 INCH))
N           ((PRINT BITS))
N           ((PAUSE))
31,03,24,34 ((START TIME))
31,03,27,34 ((END TIME))
4           ((SKIP EOF'S))
N           ((DUMP COUNTS))
!EOJ
```

```
1  UDC.PUB.ACOUSTIC          FRI. DEC  6. 1985,  1:03 PM

   !!!
   ECOPLOT INPUT,DEVICE,LIST,DUMP
   COMMENT INPUT is Intel tape name
   COMMENT DEVICE is tape drive no.
   COMMENT LIST is output device
   COMMENT DUMP is file to store echo counts
   FILE FTN10 = !INPUT,OLD;DEV=!DEVICE;REC=4500,1,U,ASCII;ACC=IN
   FILE FTN11 = WIDE,OLD
   FILE FTN15 = !DUMP
   FILE FTN20 ; DEV=!LIST
   RUN ECOPLOT/UDC.PROG
   RESET FTN10
   RESET FTN11
   RESET FTN15
   RESET FTN20
   !!!
```

```
$CONTROL FILE=10-20.NOLIST,NOSOURCE
      PROGRAM ACOUSTIC
C
C-----------------------------------------------------
C
C  THIS TEXT IS STORED AS ECOPLOT.SOURCE
C
C  A PROGRAM TO READ INTEL ACOUSTIC SYSTEM TAPES AND  PRINT BUFFER
C  HEADERS AND DATA AS BITS OR WORDS AND PLOT ECHO LEVELS BY DEPTH
C  AND TRANSMISSION
C
C-----------------------------------------------------
C
      INTEGER FNUM,BUFNUM,OUTSIZE,NUMPRINT,YEAR,DAY,HOUR,MIN,SEC,
     &        START,WIDE(1200),NARROW(1200),NUMDUMP,
     &        IBUFF(200)
      REAL TIME(1200)
      LOGICAL PAUSE,COUNTS,DATES,BUFSIZE,BUFFER(4500),BITS,DUMP,
     &        INSIDE,ECHOPLOT
      SYSTEM INTRINSIC FGETINFO, FREAD, PRINTFILEINFO
      DOUBLE PRECISION TIMESTART,TIMEEND,TIMEBUF
C
      COMMON /BUFF1/ IN,OUTSIZE,NUMPRINT,BUFNUM,BUFSIZE,ECHOPLOT,SIZE,
     &               COUNTS,PAUSE,DATES,TIMESTART,TIMEEND,IBUFF,
     &       BITS,START,BUFFER,LEN,YEAR,DAY,HOUR,MIN,SEC,NUMTIMES,
     &       TIME,WIDE,NARROW,JUMP,DUMP,NUMDUMP,TOP,BOTTOM,X,Y,XINCHES
C
      INSIDE = .FALSE.
   11 CALL INITIALIZE
C
C     JUMP PAST 'EOF' MARKS IF REQUIRED
      IF ( JUMP ) 20, 20, 15
   15 WRITE(20,14) JUMP
   14 FORMAT(" ",I3," EOF marks skipped before processing.")
      DO 18 J = 1, JUMP
   18 CALL UNITCONTROL (10,3)
C
C     READ IN AN ENTIRE BUFFER
   20 LEN = FREAD (IN,BUFFER,4500)
      IF (.CC.) 30,40,50
C
C     IF READ ERROR
   30 CALL PRINTFILEINFO (IN)
      STOP
C
C     BEGIN PROCESSING BY REVERSING ALL BITS IN EACH BYTE
   40 CALL REVERSEBITS
      BUFNUM = BUFNUM + 1
C
C     CHECK TO SEE IF BUFFER IS WITHIN TIME WINDOW
      CALL GETDATE
      TIMEBUF = DAY*1000000.0 + HOUR*10000.0 + MIN*100.0 + SEC
      IF (TIMEBUF .GT. TIMEEND)  GO TO 70
```

```
      IF (TIMEBUF .LT. TIMESTART) GO TO 20
      IF (INSIDE) GO TO 45
      INSIDE = .TRUE.
      WRITE(20,47)
   47 FORMAT(' Beginning of time window on file.')
C
C     PERFORM TASKS AS REQUIRED
   45 IF (BUFSIZE) CALL PRINTLENGTH
      IF (DATES)   CALL PRINTDATE
      IF (COUNTS.OR.BITS)  CALL PRINTCOUNTS
      IF(ECHOPLOT) CALL PLOTCOUNTS
      IF (DUMP)    CALL DUMPCOUNTS
      IF (PAUSE)   GO TO 11
      GO TO 20
C
   50 WRITE(20,60)
   60 FORMAT(' EOF MARK on input file encountered... Continuing')
      GO TO 20
   70 WRITE(20,72)
   72 FORMAT(' End of Time Window. Normal program termination')
      CALL PLOTS(IBUFF,200)
      CALL PLOT(XINCHES+10.0,0.0,999)
      STOP
      END
C
      SUBROUTINE REVERSEBITS
C
C-----------------------------------------------
C
C     THIS SUBROUTINE REVERSES THE ORDER OF ALL 8 BITS
C     IN EACH BYTE (2 BYTES/WORD) OF THE INPUT BUFFER.
C
C-----------------------------------------------
C
      INTEGER FNUM,BUFNUM,OUTSIZE,NUMPRINT,YEAR,DAY,HOUR,MIN,SEC,
     &        START,WIDE(1200),NARROW(1200),NUMDUMP,
     &        IBUFF(200)
      REAL TIME(1200)
      LOGICAL TEMP,PAUSE,COUNTS,DATES,BUFSIZE,BUFFER(4500),BITS,DUMP,
     &        ECHOPLOT
      DOUBLE PRECISION TIMESTART,TIMEEND
C
      COMMON /BUFF1/ IN,OUTSIZE,NUMPRINT,BUFNUM,BUFSIZE,ECHOPLOT,SIZE,
     &               COUNTS,PAUSE,DATES,TIMESTART,TIMEEND,IBUFF,
     &       BITS,START,BUFFER,LEN,YEAR,DAY,HOUR,MIN,SEC,NUMTIMES,
     &       TIME,WIDE,NARROW,JUMP,DUMP,NUMDUMP,TOP,BOTTOM,X,Y,XINCHES
      DO 20 IWORD = 1, LEN
      TEMP(0:1)  = BUFFER(IWORD)(7:1)
      TEMP(1:1)  = BUFFER(IWORD)(6:1)
      TEMP(2:1)  = BUFFER(IWORD)(5:1)
      TEMP(3:1)  = BUFFER(IWORD)(4:1)
      TEMP(4:1)  = BUFFER(IWORD)(3:1)
      TEMP(5:1)  = BUFFER(IWORD)(2:1)
      TEMP(6:1)  = BUFFER(IWORD)(1:1)
      TEMP(7:1)  = BUFFER(IWORD)(0:1)
      TEMP(8:1)  = BUFFER(IWORD)(15:1)
      TEMP(9:1)  = BUFFER(IWORD)(14:1)
      TEMP(10:1) = BUFFER(IWORD)(13:1)
      TEMP(11:1) = BUFFER(IWORD)(12:1)
      TEMP(12:1) = BUFFER(IWORD)(11:1)
      TEMP(13:1) = BUFFER(IWORD)(10:1)
      TEMP(14:1) = BUFFER(IWORD)(9:1)
      TEMP(15:1) = BUFFER(IWORD)(8:1)
      BUFFER(IWORD) = TEMP
   20 CONTINUE
      RETURN
      END
```

```
121   C
122         SUBROUTINE INITIALIZE
123   C
124   C-----------------------------------------------------------
125   C
126   C     THIS SUBROUTINE INITIALIZES THE INPUT & OUTPUT FILES
127   C     AND PROMPTS THE USER FOR INSTRUCTIONS TO PROCESS
128   C     THE INPUT DATA.
129   C
130   C-----------------------------------------------------------
131   C
132         INTEGER FNUM,BUFNUM,OUTSIZE,NUMPRINT,YEAR,DAY,HOUR,MIN,SEC,
133        &        START,WIDE(1200),NARROW(1200),NUMDUMP,
134        &        IBUFF(200)
135         REAL TIME(1200)
136         LOGICAL PAUSE,COUNTS,DATES,BUFSIZE,BUFFER(4500),BITS,DUMP,ECHOPLOT
137         SYSTEM INTRINSIC FGETINFO
138         DOUBLE PRECISION TIMESTART,TIMEEND
139   C
140         COMMON /BUFF1/ IN,OUTSIZE,NUMPRINT,BUFNUM,BUFSIZE,ECHOPLOT,SIZE,
141        &               COUNTS,PAUSE,DATES,TIMESTART,TIMEEND,IBUFF,
142        &       BITS,START,BUFFER,LEN,YEAR,DAY,HOUR,MIN,SEC,NUMTIMES,
143        &       TIME,WIDE,NARROW,JUMP,DUMP,NUMDUMP,TOP,BOTTOM,X,Y,XINCHES
144   C
145   C     IN   = MPE FILE NUMBER
146      10 IN   = FNUM(10)
147         CALL FGETINFO(FNUM(20),,,,OUTSIZE)
148         OUTSIZE = IABS (OUTSIZE)
149         NUMPRINT = OUTSIZE / 21
150         BUFNUM = 0
151      11 BUFSIZE = .FALSE.
152         COUNTS  = .FALSE.
153         ECHOPLOT = .FALSE.
154         PAUSE   = .FALSE.
155         DATES   = .FALSE.
156         BITS    = .FALSE.
157         DUMP    = .FALSE.
158         DISPLAY "Print Buffer sizes (Y/N)"
159         CALL YESNO ($1,92)
160          1 BUFSIZE = .TRUE.
161          2 DISPLAY "Print Dates (Y/N)"
162            CALL YESNO ($3,$4)
163          3 DATES = .TRUE.
164          4 DISPLAY "Print individual echo counts (Y/N)"
165            CALL YESNO ($5,$6)
166          5 COUNTS = .TRUE.
167          6  DISPLAY "Plot individual echos (Y/N)"
168            CALL YESNO($12,$13)
169         12  ECHOPLOT = .TRUE.
170         13  IF(ECHOPLOT) GO TO 30
171             GO TO 40
172         30  CALL PLOTS(IBUFF,200)
173             X = 0.0
```

```
174           Y = 31.0
175        40 DISPLAY "Enter top and bottom meters of layer to be plotted."
176           ACCEPT TOP,BOTTOM
177           DISPLAY "Enter size of echo trace in mv. per 1/2 inch"
178           ACCEPT SIZE
179           DISPLAY "Print individual bits in each word (Y/N)"
180           CALL YESNO (%22,%23)
181        22 BITS = .TRUE.
182        23 DISPLAY "Pause after each buffer for instructions (Y/N)"
183           CALL YESNO (%7,%20)
184         7 PAUSE = .TRUE.
185        20 DISPLAY "Time frame for data to be processed:"
186           DISPLAY "Processing begins after: (Day, hour, min., sec.) "
187           ACCEPT SDAY, SHOUR, SMIN, SSEC
188           DISPLAY "Processing  ends  after: (Day, hour, min., sec.) "
189           ACCEPT EDAY, EHOUR, EMIN, ESEC
190           TIMESTART = SDAY*1000000.0 + SHOUR*10000.0 + SMIN*100.0 + SSEC
191           TIMEEND   = EDAY*1000000.0 + EHOUR*10000.0 + EMIN*100.0 + ESEC
192           DISPLAY "Enter the # of EOF marks to jump past before"
193           DISPLAY " processing (If none, enter 0 ) "
194           ACCEPT JUMP
195           DISPLAY "Dump echo counts to device FTN15 (Y/N) "
196           CALL YESNO (%8,%9)
197         8 DUMP = .TRUE.
198           CALL FGETINFO (FNUM(15),,,,NUMDUMP)
199           NUMDUMP = IABS(NUMDUMP) / 19
200         9 CONTINUE
201           RETURN
202           END
203     C
204           SUBROUTINE PRINTLENGTH
205     C
206     C-------------------------------------------------------
207     C
208     C     THIS SUBROUTINE PRINTS THE LENGTH OF THE INPUT BUFFER
209     C     AS CALCULATED BY THE MPE INTRINSIC "FREAD" DURING THE
210     C     READ OPERATION.  THIS LENGTH IS A FUNCTION OF THE
211     C     FILE CHARACTERISTICS OF THE INPUT FILE DECLARATION.
212     C
213     C-------------------------------------------------------
214     C
215           INTEGER FNUM,BUFNUM,OUTSIZE,NUMPRINT,YEAR,DAY,HOUR,MIN,SEC,
216          &        START,WIDE(1200),NARROW(1200),NUMDUMP,
217          &        IBUFF(200)
218           REAL TIME(1200)
219           LOGICAL PAUSE,COUNTS,DATES,BUFSIZE,BUFFER(4500),BITS,DUMP,ECHOPLOT
220           DOUBLE PRECISION TIMESTART,TIMEEND
221     C
222           COMMON /BUFF1/ IN,OUTSIZE,NUMPRINT,BUFNUM,BUFSIZE,ECHOPLOT,SIZE,
223          &               COUNTS,PAUSE,DATES,TIMESTART,TIMEEND,IBUFF,
224          &       BITS,START,BUFFER,LEN,YEAR,DAY,HOUR,MIN,SEC,NUMTIMES,
225          &       TIME,WIDE,NARROW,JUMP,DUMP,NUMDUMP,TOP,BOTTOM,X,Y,XINCHES
226           WRITE(20,21) BUFNUM, LEN
227        21 FORMAT(" Buffer",I4," is",I5," words long.")
228           RETURN
229           END
230     C
231           SUBROUTINE PRINTDATE
232     C
233     C-------------------------------------------------------
234     C
235     C     THIS SUBROUTINE PRINTS THE DATE AND TIME INFORMATION
236     C     WHICH IS STORED IN THE FIRST 6 BYTES OF EACH BUFFER.
237     C
238     C-------------------------------------------------------
239     C
240           INTEGER FNUM,BUFNUM,OUTSIZE,NUMPRINT,YEAR,DAY,HOUR,MIN,SEC,
241          &        START,WIDE(1200),NARROW(1200),NUMDUMP,
242          &        IBUFF(200)
243           REAL TIME(1200)
244           LOGICAL PAUSE,COUNTS,DATES,BUFSIZE,BUFFER(4500),BITS,DUMP,ECHOPLOT
245           DOUBLE PRECISION TIMESTART,TIMEEND
246     C
```

```
      COMMON /BUFF1/ IN,OUTSIZE,NUMPRINT,BUFNUM,BUFSIZE,ECHOPLOT,SIZE,
     &              COUNTS,PAUSE,DATES,TIMESTART,TIMEEND,IBUFF,
     &       BITS,START,BUFFER,LEN,YEAR,DAY,HOUR,MIN,SEC,NUMTIMES,
     &       TIME,WIDE,NARROW,JUMP,DUMP,NUMDUMP,TOP,BOTTOM,X,Y,XINCHES
      CALL GETDATE
      WRITE(20,22) YEAR, DAY, HOUR, MIN, SEC
   22 FORMAT(' Date: 19',I2,' Julian day',I4,' Time',I3,':',
     &I2,':',I2)
      RETURN
      END
C
      SUBROUTINE PRINTCOUNTS
C
C-----------------------------------------------------------
C
C     THIS SUBROUTINE PRINTS THE INDIVIDUAL BITS STORED IN
C     EACH BYTE OF THE BUFFER (EXCLUDING THE 1ST 6 BYTES)
C     AND/OR THE INDIVIDUAL TARGET STRENGTHS (MILLIVOLTS) OF
C     EACH DIGITIZED SAMPLE OF THE TRANSMISSION FROM THE
C     ECHO SOUNDER. THE PRINTOUT WILL FILL AS MANY COLUMNS
C     OF THE OUTPUT FILE RECORD AS POSSIBLE.
C
C-----------------------------------------------------------
C
      INTEGER FNUM,BUFNUM,OUTSIZE,NUMPRINT,YEAR,DAY,HOUR,MIN,SEC,
     &       START,WIDE(1200),NARROW(1200),NUMDUMP,
     &       IBUFF(200)
      REAL TIME(1200)
      LOGICAL PAUSE,COUNTS,DATES,BUFSIZE,BUFFER(4500),BITS,DUMP,ECHOPLOT
      DOUBLE PRECISION TIMESTART,TIMEEND
C
      COMMON /BUFF1/ IN,OUTSIZE,NUMPRINT,BUFNUM,BUFSIZE,ECHOPLOT,SIZE,
     &              COUNTS,PAUSE,DATES,TIMESTART,TIMEEND,IBUFF,
     &       BITS,START,BUFFER,LEN,YEAR,DAY,HOUR,MIN,SEC,NUMTIMES,
     &       TIME,WIDE,NARROW,JUMP,DUMP,NUMDUMP,TOP,BOTTOM,X,Y,XINCHES
C
      START = 4
   10 IST = START-1
      CALL GETECHO
      IF (BITS) WRITE(20,95)
     & ( BUFFER(IST+J)(0:1),BUFFER(IST+J)(1:1),BUFFER(IST+J)(2:1),
     &   BUFFER(IST+J)(3:1),BUFFER(IST+J)(4:1),BUFFER(IST+J)(5:1),
     &   BUFFER(IST+J)(6:1),BUFFER(IST+J)(7:1),BUFFER(IST+J)(8:1),
     &   BUFFER(IST+J)(9:1),BUFFER(IST+J)(10:1),BUFFER(IST+J)(11:1),
     &   BUFFER(IST+J)(12:1),BUFFER(IST+J)(13:1),BUFFER(IST+J)(14:1),
     &   BUFFER(IST+J)(15:1) , J = 1, NUMTIMES*3 )
   95 FORMAT((' ',3(16I2,3X)))
C
      IF (.NOT.COUNTS) GO TO 45
      WRITE(20,22)
   22 FORMAT(' Time  Wide  Narrow')
      DO 45 IW = 1, NUMTIMES, NUMPRINT
      NUMOUT = MIN0 (NUMPRINT,NUMTIMES+1-IW)
      WRITE(20,23) (TIME(IW+J),WIDE(IW+J),NARROW(IW+J),J=0,NUMOUT-1)
   23 FORMAT(' ',50(F6.1,2I6,3X))
   45 CONTINUE
C
      IF (START-LEN) 10,20,20
   20 RETURN
      END
```

```
C
      SUBROUTINE GETECHO
C
C-----------------------------------------------------
C
C     THIS SUBROUTINE CONVERTS THE BINARY CODED VALUES FOR
C     TIME AND TARGET STRENGTHS INTO INTEGER VALUES.
C
C-----------------------------------------------------
C
      INTEGER FNUM,BUFNUM,OUTSIZE,NUMPRINT,YEAR,DAY,HOUR,MIN,SEC,
     &        START,WIDE(1200),NARROW(1200),NUMDUMP,
     &        IBUFF(200)
      REAL TIME(1200)
      LOGICAL PAUSE,COUNTS,DATES,BUFSIZE,BUFFER(4500),BITS,DUMP,
     &        TEMP1,TEMP2,ECHOPLOT,ENDBUF
      DOUBLE PRECISION TIMESTART,TIMEEND
C
      COMMON /BUFF1/ IN,OUTSIZE,NUMPRINT,BUFNUM,BUFSIZE,ECHOPLOT,SIZE,
     &               COUNTS,PAUSE,DATES,TIMESTART,TIMEEND,IBUFF,
     &       BITS,START,BUFFER,LEN,YEAR,DAY,HOUR,MIN,SEC,NUMTIMES,
     &       TIME,WIDE,NARROW,JUMP,DUMP,NUMDUMP,TOP,BOTTOM,X,Y,XINCHES
C
      II = 0
      NN = 0
      ENDBUF = .FALSE.
      OLDDEPTH = 0.0
      DO 10 J = START, LEN-2, 3
      II = II + 1
      DEPTH = INT(BUFFER(J)[4:12])
      DEPTH = DEPTH * .075
      IF(DEPTH.LT.TOP) GO TO 20
      IF(DEPTH.GT.BOTTOM) GO TO 20
      NN = NN + 1
      TIME(NN) = DEPTH
      TEMP1 = .NOT. BUFFER(J+1)
      TEMP2 = .NOT. BUFFER(J+2)
      WIDE(NN)  = INT(TEMP1[4:12])
      NARROW(NN) = INT(TEMP2[4:12])
C     CONVERT A/D VALUE TO MILLIVOLTS
      WIDE(NN) = WIDE(NN) * 2.44
      NARROW(NN) = NARROW(NN) * 2.44
C     APPLY 40LOG R GAIN TO WIDE BEAM AND ADJUST LEVEL SO THAT
C     5000 mv. AT 100m.DO NOT OVERFLOW OUTPUT SPACE
      TEMP = WIDE(NN) * TIME(NN) / 50.
      WIDE(NN) = TEMP + .5
      WRITE(11,21) WIDE(NN)
   21 FORMAT(I6)
   20 IF (II-1) 10,10,30
   30 IF(DEPTH-OLDDEPTH) 40,40,50
   50 OLDDEPTH = DEPTH
   10 CONTINUE
      ENDBUF = .TRUE.
      II = II + 1
      NN = NN + 1
   40 NUMINT = II - 1
      START = START + NUMINT*3
      NUMTIMES = NN
      IF(ENDBUF) NUMTIMES = NN - 1
      RETURN
      END
```

```
C
      SUBROUTINE GETDATE
C
C-----------------------------------------------------------
C
C     THIS SUBROUTINE CONVERTS THE DATE AND TIME INFORMATION
C     IN THE 1ST 6 BYTES OF A BUFFER INTO INTEGER VALUES.
C
C-----------------------------------------------------------
C
      INTEGER FNUM,BUFNUM,OUTSIZE,NUMPRINT,YEAR,DAY,HOUR,MIN,SEC,
     &        START,WIDE(1200),NARROW(1200),NUMDUMP,
     &        IBUFF(200)
      REAL TIME(1200)
      LOGICAL PAUSE,COUNTS,DATES,BUFSIZE,BUFFER(4500),BITS,DUMP,ECHOPLOT
      DOUBLE PRECISION TIMESTART,TIMEEND
C
      COMMON /BUFF1/ IN,OUTSIZE,NUMPRINT,BUFNUM,BUFSIZE,ECHOPLOT,SIZE,
     &               COUNTS,PAUSE,DATES,TIMESTART,TIMEEND,IBUFF,
     &       BITS,START,BUFFER,LEN,YEAR,DAY,HOUR,MIN,SEC,NUMTIMES,
     &       TIME,WIDE,NARROW,JUMP,DUMP,NUMDUMP,TOP,BOTTOM,X,Y,XINCHES
      YEAR = INT(BUFFER(3)[8:4])*10 + INT(BUFFER(3)[12:4])
      DAY  = INT(BUFFER(3)[4:4])*100 + INT(BUFFER(2)[8:4])*10
     &       + INT(BUFFER(2)[12:4])
      HOUR = INT(BUFFER(2)[0:4])*10 + INT(BUFFER(2)[4:4])
      MIN  = INT(BUFFER(1)[8:4])*10 + INT(BUFFER(1)[12:4])
      SEC  = INT(BUFFER(1)[0:4])*10 + INT(BUFFER(1)[4:4])
      RETURN
      END
C
      SUBROUTINE DUMPCOUNTS
C
C-----------------------------------------------------------
C
C     THIS SUBROUTINE PRINTS
C     THE INDIVIDUAL TARGET STRENGTHS (MILLIVOLTS) OF
C     EACH DIGITIZED SAMPLE OF THE TRANSMISSION FROM THE
C     ECHO SOUNDER. THE PRINTOUT WILL FILL AS MANY COLUMNS
C     OF THE OUTPUT FILE RECORD AS POSSIBLE.
C
C-----------------------------------------------------------
C
      INTEGER FNUM,BUFNUM,OUTSIZE,NUMPRINT,YEAR,DAY,HOUR,MIN,SEC,
     &        START,WIDE(1200),NARROW(1200),NUMDUMP,
     &        IBUFF(200)
      REAL TIME(1200)
      LOGICAL PAUSE,COUNTS,DATES,BUFSIZE,BUFFER(4500),BITS,DUMP,ECHOPLOT
      DOUBLE PRECISION TIMESTART,TIMEEND
C
      COMMON /BUFF1/ IN,OUTSIZE,NUMPRINT,BUFNUM,BUFSIZE,ECHOPLOT,SIZE,
     &               COUNTS,PAUSE,DATES,TIMESTART,TIMEEND,IBUFF,
     &       BITS,START,BUFFER,LEN,YEAR,DAY,HOUR,MIN,SEC,NUMTIMES,
     &       TIME,WIDE,NARROW,JUMP,DUMP,NUMDUMP,TOP,BOTTOM,X,Y,XINCHES
C
      START = 4
      CALL GETDATE
      WRITE(15,12) YEAR, DAY, HOUR, MIN, SEC
   12 FORMAT(' ',I3,I4,3I3)
   10 CALL GETECHO
      WRITE(15,13) NUMTIMES
   13 FORMAT(' ',I5)
      DO 45 IW = 1, NUMTIMES, NUMDUMP
      NUMOUT = MIN0 (NUMDUMP,NUMTIMES+1-IW)
      WRITE(15,23) (TIME(IW+J),WIDE(IW+J),NARROW(IW+J),J=0,NUMOUT-1)
   23 FORMAT(' ',50(F7.1,2I6))
   45 CONTINUE
C
      IF (START-LEN) 10,20,20
   20 RETURN
      END
```

```
C
      SUBROUTINE YESNO(*,*)
C
C-----------------------------------------------------------------
C
C     THIS SUBROUTINE RETURNS A LINE LABEL IN RESPONSE TO (Y/N)
C
C-----------------------------------------------------------------
C
      CHARACTER ANSWER*1
   10 ANSWER = ' '
      ACCEPT ANSWER
      IF(ANSWER.EQ.'Y'.OR.ANSWER.EQ.'y') RETURN 1
      IF(ANSWER.EQ.'N'.OR.ANSWER.EQ.'n'.OR.ANSWER.EQ.' ') RETURN 2
      DISPLAY 'Unable to recognize your answer'
      GO TO 10
      END
C
      SUBROUTINE PLOTCOUNTS
C
C-----------------------------------------------------------------
C
C     THIS SUBROUTINE PLOTS INDIVIDUAL INTERRUPTS AT SIZE MV = .5 INCH
C    BY DEPTH AT 1 METER = .5 INCH
C
C-----------------------------------------------------------------
C
      INTEGER FNUM,BUFNUM,OUTSIZE,NUMPRINT,YEAR,DAY,HOUR,MIN,SEC,
     &        START,WIDE(1200),NARROW(1200),NUMDUMP,
     &        IBUFF(200)
      REAL TIME(1200)
      LOGICAL PAUSE,COUNTS,DATES,BUFSIZE,BUFFER(4500),BITS,DUMP,
     &        ECHOPLOT
      DOUBLE PRECISION TIMESTART,TIMEEND
C
      COMMON /BUFF1/ IN,OUTSIZE,NUMPRINT,BUFNUM,BUFSIZE,ECHOPLOT,SIZE,
     &              COUNTS,PAUSE,DATES,TIMESTART,TIMEEND,IBUFF,
     &       BITS,START,BUFFER,LEN,YEAR,DAY,HOUR,MIN,SEC,NUMTIMES,
     &       TIME,WIDE,NARROW,JUMP,DUMP,NUMDUMP,TOP,BOTTOM,X,Y,XINCHES
      IF(Y.EQ.31.0) GO TO 40
      Y = Y-.5
      CALL PLOT(X,Y,3)
      CALL PLOT(X+.1,Y,2)
      GO TO 50
   40 CALL AXES
   50 START = 4
   10 CALL GETECHO
      IF(Y.LT.1.0) GO TO 60
      GO TO 70
   60 X = X +(BOTTOM-TOP)/2.0 +1.0
      CALL AXES
   70 Y = Y-.5
      DO 45 IW = 1,NUMTIMES
      DEPTH = TIME(IW)
      XINCHES = X + (BOTTOM-DEPTH)/2.0
      YINCHES = Y -.25 *WIDE(IW)/SIZE
      CALL PLOT(XINCHES,YINCHES,3)
      YINCHES = Y +.25 *WIDE(IW)/SIZE
      CALL PLOT(XINCHES,YINCHES,2)
   45 CONTINUE
      IF (START-LEN) 10,20,20
   20 RETURN
      END
```

```
C
      SUBROUTINE AXES
C
C-----------------------------------------------------------
C
C     THIS SUBROUTINE PLOTS AXES AND LABELS DEPTH AXIS IN METERS
C
C-----------------------------------------------------------
C
      INTEGER FNUM,BUFNUM,OUTSIZE,NUMPRINT,YEAR,DAY,HOUR,MIN,SEC,
     &        START,WIDE(1200),NARROW(1200),NUMDUMP,
     &        IBUFF(200)
      REAL TIME(1200)
      LOGICAL PAUSE,COUNTS,DATES,BUFSIZE,BUFFER(4500),BITS,DUMP,
     &        ECHOPLOT
      DOUBLE PRECISION TIMESTART,TIMEEND
      COMMON /BUFF1/ IN,OUTSIZE,NUMPRINT,BUFNUM,BUFSIZE,ECHOPLOT,SIZE,
     &               COUNTS,PAUSE,DATES,TIMESTART,TIMEEND,IBUFF,
     &       BITS,START,BUFFER,LEN,YEAR,DAY,HOUR,MIN,SEC,NUMTIMES,
     &       TIME,WIDE,NARROW,JUMP,DUMP,NUMDUMP,TOP,BOTTOM,X,Y,XINCHES
      XLENGTH = BOTTOM-TOP
      CALL PLOT(X,0.0,3)
      CALL PLOT(X,31.0,2)
      TIC = X
      DO 25 I = 1,XLENGTH
      TIC = TIC+.5
      CALL PLOT(TIC,31.0,2)
      CALL PLOT(TIC,30.9,2)
      CALL PLOT(TIC,31.0,2)
   25 CONTINUE
      TIC = X
      STEP = 0.0
      DO 35 K = 1,XLENGTH
      CALL NUMBER(TIC,31.5,.1,(BOTTOM-STEP),270.0,0)
      TIC = TIC+.5
      STEP = STEP +1.0
   35 CONTINUE
      Y = 31.0
      RETURN
      END
```